SQL : requêtes simples

Thibaud Lemanissier

PC Lycée Carnot

2026-2027

Comment gérer des données

Il existe beaucoup de méthodes pour stoker des informations :

- un fichier texte (fichier .txt, .docx,...) ;
- un tableur (fichier .xlsx, .odt,...) ;
- ...

Comment gérer des données

Il existe beaucoup de méthodes pour stoker des informations :

- un fichier texte (fichier .txt, .docx,...) ;
- un tableur (fichier .xlsx, .odt,...) ;
- ...

Si notre objectif est de stoker et gérer une base de donnée, il faut pouvoir facilement :

- accéder une partie de ces données ;
- stoker efficacement (en limitant les répétitions) ;
- ajouter/supprimer des données ;
- ...

Pour cela nous allons utiliser les **bases données relationnelles**.

Comment gérer des données

Il existe beaucoup de méthodes pour stoker des informations :

- un fichier texte (fichier .txt, .docx,...) ;
- un tableur (fichier .xlsx, .odt,...) ;
- ...

Si notre objectif est de stoker et gérer une base de donnée, il faut pouvoir facilement :

- accéder une partie de ces données ;
- stoker efficacement (en limitant les répétitions) ;
- ajouter/supprimer des données ;
- ...

Pour cela nous allons utiliser les **bases données relationnelles**.
Notre objectif va être de comprendre comment procéder pour le premier point.
Pour cela, nous allons utiliser le langage **SQL** (Structured Query Language).

L'exemple des communes de France

Nous allons nous intéresser aux communes de France :

id	Nom communes	id dep	nom dep	id reg	nom reg	pop
1001	L'Abergement-Clémenciat	1	Ain	82	Rhône-Alpes	784
1002	L'Abergement-de-Varey	1	Ain	82	Rhône-Alpes	221
1004	Ambérieu-en-Bugey	1	Ain	82	Rhône-Alpes	13835
1005	Ambérieux-en-Dombes	1	Ain	82	Rhône-Alpes	1616
1006	Ambléon	1	Ain	82	Rhône-Alpes	116
1007	Ambronay	1	Ain	82	Rhône-Alpes	2362
1008	Ambutrix	1	Ain	82	Rhône-Alpes	729
1009	Andert-et-Condon	1	Ain	82	Rhône-Alpes	340
1010	Anglefort	1	Ain	82	Rhône-Alpes	994
1011	Apremont	1	Ain	82	Rhône-Alpes	363
1012	Aranc	1	Ain	82	Rhône-Alpes	302
1013	Arandas	1	Ain	82	Rhône-Alpes	163
1014	Arbent	1	Ain	82	Rhône-Alpes	3476
1015	Arbignieu	1	Ain	82	Rhône-Alpes	480
1016	Arbigny	1	Ain	82	Rhône-Alpes	399
1017	Argis	1	Ain	82	Rhône-Alpes	421
1019	Armix	1	Ain	82	Rhône-Alpes	20

On remarque qu'il y a de nombreuses informations redondantes :

- Si on connait le numéro du département, on peut obtenir le nom du département ;
- Si on connait le nom du département, on peut obtenir le nom de la région ;
- ...

L'exemple des communes de France

Pour diminuer la place prise par ce tableau et pour facilité les éventuelles corrections/modifications ce tableau a été divisé en trois tableaux : communes, departements, regions

id	dep	nom	pop
1001	1	L'Abergement-Clémenciat	784
1002	1	L'Abergement-de-Varey	221
1004	1	Ambérieu-en-Bugey	13835
1005	1	Ambérieux-en-Dombes	1616
1006	1	Ambléon	116
1007	1	Ambronay	2362
1008	1	Ambutrix	729
1009	1	Andert-et-Condon	340
1010	1	Anglefort	994
1011	1	Apremont	363

id	reg	nom
971	1	Guadeloupe
972	2	Martinique
973	3	Guyane
974	4	La Réunion
976	6	Mayotte
75	11	Paris
95	11	Val-d'Oise
94	11	Val-de-Marne
93	11	Seine-Saint-Denis
92	11	Hauts-de-Seine

id	nom
1	Guadeloupe
2	Martinique
3	Guyane
4	La Réunion
6	Mayotte
11	Île-de-France
21	Champagne-Ardenne
22	Picardie
23	Haute-Normandie
24	Centre

Structure de la table

table, attribues et enregistrements

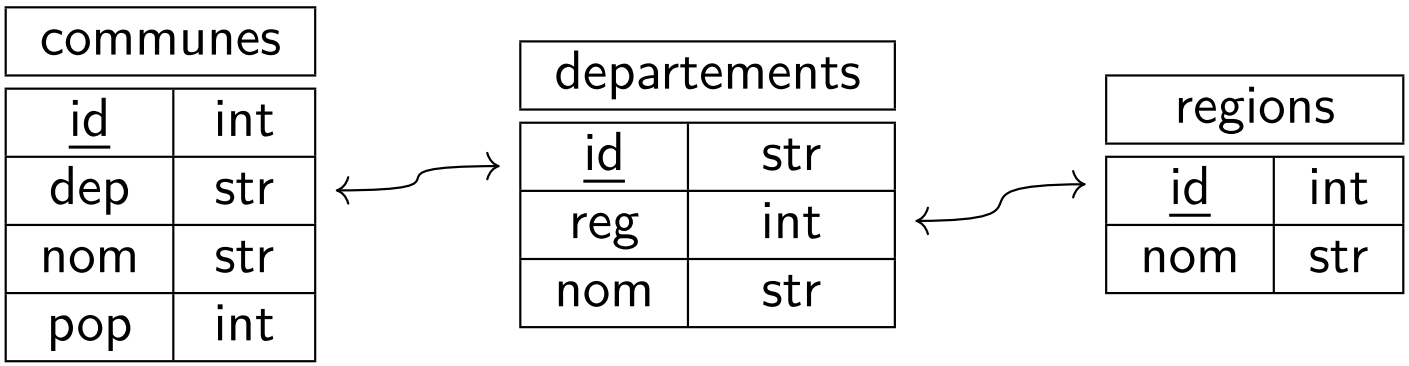
Chacun des tableaux sera appelé **table**, dont les colonnes seront appelées **attribues** et les lignes seront appelées **enregistrements**. L'ensemble des tables sera **la base de donnée**. Le domaine d'un **attribut** est l'ensemble des valeurs que peut prendre cet attribut.

Structure de la table

table, attribues et enregistrements

Chacun des tableaux sera appelé **table**, dont les colonnes seront appelées **attribues** et les lignes seront appelées **enregistrements**. L'ensemble des tables sera **la base de donnée**. Le domaine d'un **attribut** est l'ensemble des valeurs que peut prendre cet attribut.

Pour synthétiser ce que contient la base de données (les attributs et leur domaine) ainsi que le lien entre les différentes tables, on peut le représenter de la manière suivante :

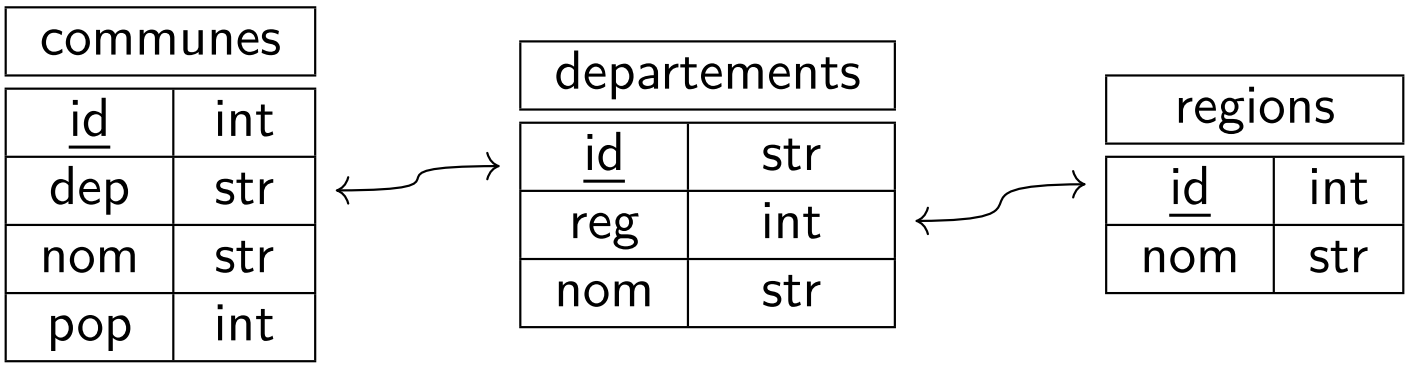


Structure de la table

table, attribues et enregistrements

Chacun des tableaux sera appelé **table**, dont les colonnes seront appelées **attribues** et les lignes seront appelées **enregistrements**. L'ensemble des tables sera **la base de donnée**. Le domaine d'un **attribut** est l'ensemble des valeurs que peut prendre cet attribut.

Pour synthétiser ce que contient la base de données (les attributs et leur domaine) ainsi que le lien entre les différentes tables, on peut le représenter de la manière suivante :



On peut remarquer que la colonne dep est de type `str`. Pourquoi ?

Clé d'une table

Clé

Une **clé primaire** est un attribut (ou un ensemble minimal d'attribut) d'une table permettant de retrouver de manière unique un enregistrement.
Une **clé étrangaire** est un attribut d'une table qui est la clé primaire d'une autre table.

Clé d'une table

Clé

Une **clé primaire** est un attribut (ou un ensemble minimal d'attribut) d'une table permettant de retrouver de manière unique un enregistrement.
Une **clé étrangaire** est un attribut d'une table qui est la clé primaire d'une autre table.

On peut synthétiser une table par son schéma, pour les tables de la base de donnée que l'on considère, on a donc :

communes(<u>id : int</u>, dep : str, nom : str, pop : int)
departements(<u>id : int</u>, reg : int, nom : str)
regions(<u>id : int</u>, nom : str)

Clé d'une table

Clé

Une **clé primaire** est un attribut (ou un ensemble minimal d'attribut) d'une table permettant de retrouver de manière unique un enregistrement.
Une **clé étrangaire** est un attribut d'une table qui est la clé primaire d'une autre table.

On peut synthétiser une table par son schéma, pour les tables de la base de donnée que l'on considère, on a donc :

communes(<u>id : int</u>, dep : str, nom : str, pop : int)
departements(<u>id : int</u>, reg : int, nom : str)
regions(<u>id : int</u>, nom : str)

Pourquoi nom n'est pas une clé de communes ?

Premières requêtes

Pour récupérer des informations de la base de donnée, on tape des **requêtes**. Le résultats d'une requête est à nouveau une table.
Pour récupérer la table de toutes les communes qui ont une population supérieure à 300000 habitants, on tape :

Premières requêtes

Pour récupérer des informations de la base de donnée, on tape des **requêtes**. Le résultats d'une requête est à nouveau une table.
Pour récupérer la table de toutes les communes qui ont une population supérieure à 300000 habitants, on tape :

```
1 SELECT * FROM communes WHERE pop >= 300000 ;
```

Premières requêtes

Pour récupérer des informations de la base de donnée, on tape des **requêtes**. Le résultats d'une requête est à nouveau une table.
Pour récupérer la table de toutes les communes qui ont une population supérieure à 300000 habitants, on tape :

```
1 SELECT * FROM communes WHERE pop >= 300000 ;
```

On obtient :

id	dep	nom	pop
6088	6	Nice	343304
13055	13	Marseille	850726
31555	31	Toulouse	441802
69123	69	Lyon	484344
75056	75	Paris	2243833

Premières requêtes

Pour récupérer des informations de la base de donnée, on tape des **requêtes**. Le résultats d'une requête est à nouveau une table.
Pour récupérer la table de toutes les communes qui ont une population supérieure à 300000 habitants, on tape :

```
1 SELECT * FROM communes WHERE pop >= 300000 ;
```

On obtient :

id	dep	nom	pop
6088	6	Nice	343304
13055	13	Marseille	850726
31555	31	Toulouse	441802
69123	69	Lyon	484344
75056	75	Paris	2243833

L'ordre des mots-clefs est important, il ne peut pas être différent de celui-ci.
Enfin, on peut écrire indifféremment les mots-clefs minuscules ou majuscules, cela n'a aucune incidence (on dit que SQL n'est pas sensible à la casse). Toute requête en SQL contient les mots clefs `SELECT` et `FROM` et est terminée par un « ; ».

Premières requêtes

On peut mettre plusieurs conditions dans une requête, pour cela on peut utiliser les mots clés `AND`, `OR`, `NOT`.
Si l'on souhaite obtenir les noms et la population des villes qui sont dans la Drôme (26) et qui ont moins de 100 habitants, on peut écrire :

Premières requêtes

On peut mettre plusieurs conditions dans une requête, pour cela on peut utiliser les mots clés `AND`, `OR`, `NOT`.
Si l'on souhaite obtenir les noms et la population des villes qui sont dans la Drôme (26) et qui ont moins de 100 habitants, on peut écrire :

```
1 SELECT nom, pop FROM communes WHERE dep = 26 AND pop <= 100 ;
```

(Vous remarquez que c'est un simple égale)

Premières requêtes

On peut mettre plusieurs conditions dans une requête, pour cela on peut utiliser les mots clés `AND`, `OR`, `NOT`.
Si l'on souhaite obtenir les noms et la population des villes qui sont dans la Drôme (26) et qui ont moins de 100 habitants, on peut écrire :

```
1 SELECT nom, pop FROM communes WHERE dep = 26 AND pop <= 100 ;
```

(Vous remarquez que c'est un simple égale)
On obtient :

nom	pop
Aulan	5
La Bâtie-des-Fonds	8
Izon-la-Bruisse	8
Laux-Montaux	1
Rochefourchat	1

Recherche de motif

On peut utiliser le mot clé `LIKE` pour rechercher un motif dans une chaine de caractères.
Dans le motif recherché, on peut faire figurer le caractère « % » correspondant à une chaine de caractères de taille quelconque et « _ » correspondant à exactement un unique caractère.
Pour chercher les communes dont le nom contient « bois », on peut écrire :

Recherche de motif

On peut utiliser le mot clé `LIKE` pour rechercher un motif dans une chaine de caractères.
Dans le motif recherché, on peut faire figurer le caractère « % » correspondant à une chaine de caractères de taille quelconque et « _ » correspondant à exactement un unique caractère.
Pour chercher les communes dont le nom contient « bois », on peut écrire :

```
1 SELECT * FROM communes WHERE nom LIKE "%bois%" ;
```

Recherche de motif

On peut utiliser le mot clé `LIKE` pour rechercher un motif dans une chaine de caractères.
Dans le motif recherché, on peut faire figurer le caractère « % » correspondant à une chaine de caractères de taille quelconque et « _ » correspondant à exactement un unique caractère.
Pour chercher les communes dont le nom contient « bois », on peut écrire :

```
1 SELECT * FROM communes WHERE nom LIKE "%bois%" ;
```

On obtient (pour les premiers enregistrements) :

id	dep	nom	pop
1049	1	La Boisse	2928
1050	1	Boissey	288
1340	1	Saint-Bois	123
1350	1	Saint-Étienne-du-Bois	2441
1444	1	Villebois	1153
2030	2	Aubencheul-aux-Bois	310
2096	2	Bois-lès-Pargny	183

Fonctions d'agrégation

On peut utiliser les fonctions `MIN`, `MAX`, `SUM`, `AVG` (pour calculer la moyenne) et `COUNT` (pour le nombre d'enregistrements) sur les attributs entre `SELECT` et `FROM`.
Pour obtenir le nombre de communes et la population totale du calvados (14) ainsi que , on peut taper :

Fonctions d'agrégation

On peut utiliser les fonctions `MIN`, `MAX`, `SUM`, `AVG` (pour calculer la moyenne) et `COUNT` (pour le nombre d'enregistrements) sur les attributs entre `SELECT` et `FROM`. Pour obtenir le nombre de communes et la population totale du calvados (14) ainsi que , on peut taper :

```
1 SELECT COUNT(*) AS Nombre_de_communes, SUM(pop) AS Population_totale →
      FROM communes WHERE dep = 14 ;
```

Le `AS` permet de donner un alias.

Fonctions d'agrégation

On peut utiliser les fonctions `MIN`, `MAX`, `SUM`, `AVG` (pour calculer la moyenne) et `COUNT` (pour le nombre d'enregistrements) sur les attributs entre `SELECT` et `FROM`.
Pour obtenir le nombre de communes et la population totale du calvados (14) ainsi que , on peut taper :

```
1 SELECT COUNT(*) AS Nombre_de_communes, SUM(pop) AS Population_totale →
      FROM communes WHERE dep = 14 ;
```

Le `AS` permet de donner un alias.
On obtient :

Nombre_de_communes	Population_totale
706	683105

Syntaxe

La précédente requête commence à être un peu dure à lire. Comme nous l'avons déjà vu, le SQL n'est pas sensible à la casse. Nous pouvons passer à la ligne sans indentation à tout moment.
Pour une plus grande clarté et de lisibilité, nous allons passer à la ligne à chaque mot structurant important.
Pour la requête précédente, nous écrirons :

Syntaxe

La précédente requête commence à être un peu dure à lire. Comme nous l'avons déjà vu, le SQL n'est pas sensible à la casse. Nous pouvons passer à la ligne sans indentation à tout moment.
Pour une plus grande clarté et de lisibilité, nous allons passer à la ligne à chaque mot structurant important.
Pour la requête précédente, nous écrirons :

```sql
SELECT COUNT(*) AS Nombre_de_communes, SUM(pop) AS Population_totale
FROM communes
WHERE dep = 14 ;
```

Fonctions d'agrégation

On peut donc utiliser utiliser les fonctions `SUM` et `COUNT` pour calculer d'une autre manière la moyenne des populations des communes dans le calvados :

Fonctions d'agrégation

On peut donc utiliser utiliser les fonctions `SUM` et `COUNT` pour calculer d'une autre manière la moyenne des populations des communes dans le calvados :

```
1 SELECT SUM(pop)/COUNT(*) AS Moyenne
2 FROM communes
3 WHERE dep = 14 ;
```

Fonctions d'agrégation

On peut donc utiliser utiliser les fonctions `SUM` et `COUNT` pour calculer d'une autre manière la moyenne des populations des communes dans le calvados :

```
1 SELECT SUM(pop)/COUNT(*) AS Moyenne
2 FROM communes
3 WHERE dep = 14 ;
```

Ce qui se voit dans l'affichage :

Moyenne
967

On remarque que puisque pop est un entier la division effectée est la division euclidienne pour obtenir un entier. On privilégiera donc `AVG`.

Ordonner et limiter l'affichage

Les mots clés `ORDER BY` permet ordonner les enregistrements de manière croissante en ajoutant `ASC` et décroissante en ajoutant `DESC`.
On peut les combiner avec les mot clé `LIMIT` pour limiter le nombre d'enregistrement et `OFFSET` pour n'afficher qu'à partir d'un certain rang. Comme en python, la numérotation commence à 0.
Par exemple, pour afficher entre la 10 et la 15 communes les plus peuplés (incluse), on peut écrire :

Ordonner et limiter l'affichage

Les mots clés `ORDER BY` permet ordonner les enregistrements de manière croissante en ajoutant `ASC` et décroissante en ajoutant `DESC`.
On peut les combiner avec les mot clé `LIMIT` pour limiter le nombre d'enregistrement et `OFFSET` pour n'afficher qu'à partir d'un certain rang. Comme en python, la numérotation commence à 0.
Par exemple, pour afficher entre la 10 et la 15 communes les plus peuplés (incluse), on peut écrire :

```
1 SELECT *
2 FROM communes
3 ORDER BY pop ASC
4 LIMIT 6
5 OFFSET 9 ;
```

Ordonner et limiter l'affichage

Les mots clés `ORDER BY` permet ordonner les enregistrements de manière croissante en ajoutant `ASC` et décroissante en ajoutant `DESC`.
On peut les combiner avec les mot clé `LIMIT` pour limiter le nombre d'enregistrement et `OFFSET` pour n'afficher qu'à partir d'un certain rang. Comme en python, la numérotation commence à 0.
Par exemple, pour afficher entre la 10 et la 15 communes les plus peuplés (incluse), on peut écrire :

```sql
SELECT *
FROM communes
ORDER BY pop ASC
LIMIT 6
OFFSET 9 ;
```

id	dep	nom	pop
59350	59	Lille	227560
35238	35	Rennes	207178
51454	51	Reims	179992
76351	76	Le Havre	175497
42218	42	Saint-Étienne	171260
83137	83	Toulon	164532

Ordonner et limiter l'affichage

Les mots clés `ORDER BY` permet ordonner les enregistrements de manière croissante en ajoutant `ASC` et décroissante en ajoutant `DESC`.
On peut les combiner avec les mot clé `LIMIT` pour limiter le nombre d'enregistrement et `OFFSET` pour n'afficher qu'à partir d'un certain rang. Comme en python, la numérotation commence à 0.
Par exemple, pour afficher entre la 10 et la 15 communes les plus peuplés (incluse), on peut écrire :

```sql
SELECT *
FROM communes
ORDER BY pop ASC
LIMIT 6
OFFSET 9 ;
```

id	dep	nom	pop
59350	59	Lille	227560
35238	35	Rennes	207178
51454	51	Reims	179992
76351	76	Le Havre	175497
42218	42	Saint-Étienne	171260
83137	83	Toulon	164532

Pour utiliser `ORDER BY`, `LIMIT` et `OFFSET`, on prendra bien garde à respecter l'ordre.

Valeurs distinctes

On peut utiliser le mot clé `DISTINCT` pour spécifier que l'on garde uniquement des enregistrements d'attribut distinct.
Pour obtenir les noms de commune distinct, on peu écrire :

Valeurs distinctes

On peut utiliser le mot clé `DISTINCT` pour spécifier que l'on garde uniquement des enregistrements d'attribut distinct.
Pour obtenir les noms de commune distinct, on peu écrire :

```
1 SELECT DISTINCT nom
2 FROM communes ;
```

Valeurs distinctes

On peut utiliser le mot clé `DISTINCT` pour spécifier que l'on garde uniquement des enregistrements d'attribut distinct.
Pour obtenir les noms de commune distinct, on peu écrire :

```
1 SELECT DISTINCT nom
2 FROM communes ;
```

On peut l'utiliser dans une fonction d'agrégation :

Valeurs distinctes

On peut utiliser le mot clé `DISTINCT` pour spécifier que l'on garde uniquement des enregistrements d'attribut distinct.
Pour obtenir les noms de commune distinct, on peu écrire :

```sql
SELECT DISTINCT nom
FROM communes ;
```

On peut l'utiliser dans une fonction d'agrégation :

```sql
SELECT COUNT(DISTINCT nom) AS nb_nom_distinct, COUNT(nom) AS nb_nom
FROM communes ;
```

Valeurs distinctes

On peut utiliser le mot clé `DISTINCT` pour spécifier que l'on garde uniquement des enregistrements d'attribut distinct.
Pour obtenir les noms de commune distinct, on peu écrire :

```sql
SELECT DISTINCT nom
FROM communes ;
```

On peut l'utiliser dans une fonction d'agrégation :

```sql
SELECT COUNT(DISTINCT nom) AS nb_nom_distinct, COUNT(nom) AS nb_nom
FROM communes ;
```

On obtient :

nb_nom_distinct	nb_nom
34154	36705

Valeurs distinctes

Écrire une requête permettant d'obtenir le nombre de nom de communes distincts en Seine Maritime (76).

Union

On peut utiliser le mot clé `UNION` pour faire l'union (non disjointe) entre les enregistrements obtenu par 2 requêtes possédant les mêmes attributs.
Pour obtenir les communes dans la Lozère (48) ou ayant une population inférieur à 10000 habitants, on peut écrire :

Union

On peut utiliser le mot clé `UNION` pour faire l'union (non disjointe) entre les enregistrements obtenu par 2 requêtes possédant les mêmes attributs.
Pour obtenir les communes dans la Lozère (48) ou ayant une population inférieur à 10000 habitants, on peut écrire :

```
1 SELECT *
2 FROM communes
3 WHERE dep = 48
4 UNION
5 SELECT *
6 FROM communes
7 WHERE pop <= 10000 ;
```

Union

On peut utiliser le mot clé `UNION` pour faire l'union (non disjointe) entre les enregistrements obtenu par 2 requêtes possédant les mêmes attributs.
Pour obtenir les communes dans la Lozère (48) ou ayant une population inférieur à 10000 habitants, on peut écrire :

```sql
SELECT *
FROM communes
WHERE dep = 48
UNION
SELECT *
FROM communes
WHERE pop <= 10000 ;
```

id	dep	nom	pop
10002	10	Ailleville	276
10003	10	Aix-en-Othe	2434
10004	10	Allibaudières	268
10005	10	Amance	275
10006	10	Arcis-sur-Aube	3011
10007	10	Arconville	113
10008	10	Argançon	100
10009	10	Arrelles	87

Intersection

On peut utiliser le mot clé `INTERSECT` pour faire l'intersection entre les enregistrements obtenu par 2 requêtes possédant les mêmes attributs.
Pour obtenir les communes dans la Lozère (48) et ayant une population supérieur à 10000 habitants, on peut écrire :

Intersection

On peut utiliser le mot clé `INTERSECT` pour faire l'intersection entre les enregistrements obtenu par 2 requêtes possédant les mêmes attributs.
Pour obtenir les communes dans la Lozère (48) et ayant une population supérieur à 10000 habitants, on peut écrire :

```
1 SELECT *
2 FROM communes
3 WHERE dep = 48
4 INTERSECT
5 SELECT *
6 FROM communes
7 WHERE pop >= 10000 ;
```

Intersection

On peut utiliser le mot clé `INTERSECT` pour faire l'intersection entre les enregistrements obtenu par 2 requêtes possédant les mêmes attributs.
Pour obtenir les communes dans la Lozère (48) et ayant une population supérieur à 10000 habitants, on peut écrire :

```
1 SELECT *
2 FROM communes
3 WHERE dep = 48
4 INTERSECT
5 SELECT *
6 FROM communes
7 WHERE pop >= 10000 ;
```

id	dep	nom	pop
48095	48	Mende	12140

Différence

On peut utiliser le mot clé `EXCEPT` pour garder les enregistrements obtenus par la première requête mais qui n'apparaissent pas dans la seconde.
Pour obtenir les communes dans la Lozère (48) mais n'ayant pas une population inférieur à 10000 habitants, on peut écrire :

Différence

On peut utiliser le mot clé `EXCEPT` pour garder les enregistrements obtenus par la première requête mais qui n'apparaissent pas dans la seconde.
Pour obtenir les communes dans la Lozère (48) mais n'ayant pas une population inférieur à 10000 habitants, on peut écrire :

```
1 SELECT *
2 FROM communes
3 WHERE dep = 48
4 EXCEPT
5 SELECT *
6 FROM communes
7 WHERE pop <= 10000 ;
```

Différence

On peut utiliser le mot clé `EXCEPT` pour garder les enregistrements obtenus par la première requête mais qui n'apparaissent pas dans la seconde.
Pour obtenir les communes dans la Lozère (48) mais n'ayant pas une population inférieur à 10000 habitants, on peut écrire :

```
1 SELECT *
2 FROM communes
3 WHERE dep = 48
4 EXCEPT
5 SELECT *
6 FROM communes
7 WHERE pop <= 10000 ;
```

id	dep	nom	pop
48095	48	Mende	12140

Reformulation

Les unions, intersections, différence peuvent aussi être reformulée avec des `OR`, `AND`, `NOT` dans le cas où la requête ne concerne qu'une table.

Produit cartésien

Si l'on souhaite avoir simultanément deux tables, on peut faire un produit cartésien.
Pour faire le produit cartésien de la table communes et la table departements, on peut écrire :

Produit cartésien

Si l'on souhaite avoir simultanément deux tables, on peut faire un produit cartésien.
Pour faire le produit cartésien de la table communes et la table departements, on peut écrire :

```
1 SELECT *
2 FROM communes, departements ;
```

Produit cartésien

Si l'on souhaite avoir simultanément deux tables, on peut faire un produit cartésien.
Pour faire le produit cartésien de la table communes et la table departements, on peut écrire :

```
1 SELECT *
2 FROM communes, departements ;
```

id	dep	nom	pop	id	reg	nom
1001	1	L'Abergement-Clémenciat	784	971	1	Guadeloupe
1001	1	L'Abergement-Clémenciat	784	972	2	Martinique
1001	1	L'Abergement-Clémenciat	784	973	3	Guyane
1001	1	L'Abergement-Clémenciat	784	974	4	La Réunion
1001	1	L'Abergement-Clémenciat	784	976	6	Mayotte
1001	1	L'Abergement-Clémenciat	784	75	11	Paris

Produit cartésien

Le problème que dans ce cas, il n'y a pas lien entre les 4 premières colonnes et les 3 dernières. Pour résoudre ce problème, on peu ajouter une condition.

Produit cartésien

Le problème que dans ce cas, il n'y a pas lien entre les 4 premières colonnes et les 3 dernières. Pour résoudre ce problème, on peu ajouter une condition.

```
1 SELECT *
2 FROM communes, departements
3 WHERE communes.dep = departements.id ;
```

Produit cartésien

Le problème que dans ce cas, il n'y a pas lien entre les 4 premières colonnes et les 3 dernières. Pour résoudre ce problème, on peu ajouter une condition.

```
1 SELECT *
2 FROM communes, departements
3 WHERE communes.dep = departements.id ;
```

Cependant, il y a une méthode plus naturelle.

Jointure

Lorsque l'on a besoin simultanément de 2 (ou plus) tables, on peut utiliser les mots-clefs `JOIN` et `ON` pour faire une jointure entre les tables.
Si on veut afficher les noms des communes avec le nom du département correspondant, on peut écrire :

Jointure

Lorsque l'on a besoin simultanément de 2 (ou plus) tables, on peut utiliser les mots-clefs `JOIN` et `ON` pour faire une jointure entre les tables.
Si on veut afficher les noms des communes avec le nom du département correspondant, on peut écrire :

```
1 SELECT c.nom AS commune, d.nom AS departement
2 FROM communes AS c JOIN departements AS d
3 ON  c.dep = d.id ;
```

Jointure

Lorsque l'on a besoin simultanément de 2 (ou plus) tables, on peut utiliser les mots-clefs `JOIN` et `ON` pour faire une jointure entre les tables.
Si on veut afficher les noms des communes avec le nom du département correspondant, on peut écrire :

```
1 SELECT c.nom AS commune, d.nom AS departement
2 FROM communes AS c JOIN departements AS d
3 ON  c.dep = d.id ;
```

On obtient (pour les premiers enregistrements) :

commune	departement
L'Abergement-Clémenciat	Ain
L'Abergement-de-Varey	Ain
Ambérieu-en-Bugey	Ain
Ambérieux-en-Dombes	Ain
Ambléon	Ain
Ambronay	Ain

Jointure

Si on veut afficher les noms des communes avec le nom du département de la région Picardie, on peut écrire :

Jointure

Si on veut afficher les noms des communes avec le nom du département de la région Picardie, on peut écrire :

```sql
SELECT c.nom AS commune, d.nom AS departement
FROM communes AS c JOIN departements AS d JOIN regions AS r
ON  c.dep = d.id AND r.id = d.reg
WHERE r.nom = "Picardie" ;
```

Jointure

Si on veut afficher les noms des communes avec le nom du département de la région Picardie, on peut écrire :

```sql
SELECT c.nom AS commune, d.nom AS departement
FROM communes AS c JOIN departements AS d JOIN regions AS r
ON  c.dep = d.id AND r.id = d.reg
WHERE r.nom = "Picardie" ;
```

On obtient (pour les premiers enregistrements) :

commune	departement
Abbécourt	Aisne
Achery	Aisne
Acy	Aisne
Agnicourt-et-Séchelles	Aisne
Aguilcourt	Aisne
Aisonville-et-Bernoville	Aisne

Auto-jointure

Il peut par moment être utilise de joindre d'une table avec elle même.
Si on veut afficher les noms des communes qui ont un homonyme dans l'Ain (1), on peut écrire :

Auto-jointure

Il peut par moment être utilise de joindre d'une table avec elle même.
Si on veut afficher les noms des communes qui ont un homonyme dans l'Ain (1), on peut écrire :

```sql
SELECT c1.nom AS commune_1, c2.dep AS departement
FROM communes AS c1 JOIN communes AS c2
ON  c1.nom = c2.nom
WHERE c1.id <> c2.id AND c1.dep = 1 ;
```

Le symbole « <> » signifie différent en SQL (attention à ne pas confondre avec Python)

Auto-jointure

Il peut par moment être utilise de joindre d'une table avec elle même.
Si on veut afficher les noms des communes qui ont un homonyme dans l'Ain (1), on peut écrire :

```sql
SELECT c1.nom AS commune_1, c2.dep AS departement
FROM communes AS c1 JOIN communes AS c2
ON  c1.nom = c2.nom
WHERE c1.id <> c2.id AND c1.dep = 1 ;
```

Le symbole « <> » signifie différent en SQL (attention à ne pas confondre avec Python)
On obtient (pour les premiers enregistrements) :

commune	departement
Apremont	60
Apremont	70
Apremont	73
Apremont	8
Apremont	85
Balan	8

Auto-jointure

Question

Écrire une requête permettant d'afficher toutes les noms de communes qui apparaissent dans des département différent. La requête devra afficher le premier département, le deuxième et le nom de la commune.