

# SQL 1 : Requêtes sur une table

## SELECT, WHERE, DISTINCT, ORDER BY, AS

Quentin Fortier

April 6, 2023

## Question

Comment gérer de façon efficace un grand nombre de données ?

## Exemples :

- 1 Les astres connus, avec leur taille, masse...
- 2 Les espèces connues, avec leur taxonomie, famille...
- 3 Les utilisateurs d'un site web, avec leur mot de passe, préférences...
- 4 Les élèves d'un lycée, avec leurs notes, options...

# Base de données : Motivation

Pour stocker des données d'élèves, on peut utiliser :

- 1 Un tableau Excel dont les lignes sont les élèves et les colonnes les attributs (nom, prénom, classe, moyenne...).
- 2 Une liste Python : `[('Tao', 'Terence', 'MP', 12), ...]`
- 3 Un dictionnaire Python :  
`{'Tao': ('Terence', 'MP', 12), ...}`

# Base de données : Motivation

Pour stocker des données d'élèves, on peut utiliser :

- ① Un tableau Excel dont les lignes sont les élèves et les colonnes les attributs (nom, prénom, classe, moyenne...).
- ② Une liste Python : `[('Tao', 'Terence', 'MP', 12), ...]`
- ③ Un dictionnaire Python :  
`{'Tao': ('Terence', 'MP', 12), ...}`

Avec un tableur ou un dictionnaire, on accède facilement aux données d'un élève mais il est difficile et lent d'accéder à tous élèves d'une classe, par exemple.

# Base de données : Motivation

Pour stocker des données d'élèves, on peut utiliser :

- 1 Un tableau Excel dont les lignes sont les élèves et les colonnes les attributs (nom, prénom, classe, moyenne...).
- 2 Une liste Python : `[('Tao', 'Terence', 'MP', 12), ...]`
- 3 Un dictionnaire Python :  
`{'Tao': ('Terence', 'MP', 12), ...}`

Avec un tableur ou un dictionnaire, on accède facilement aux données d'un élève mais il est difficile et lent d'accéder à tous élèves d'une classe, par exemple.

→ On utilise plutôt une **base de donnée**.

## Base de donnée

Une **base de donnée** est un ensemble de **tables**. On peut y extraire des informations à l'aide de **requêtes**.

## Base de donnée

Une **base de donnée** est un ensemble de **tables**. On peut y extraire des informations à l'aide de **requêtes**.

## Table

Une **table** est un tableau à 2 dimensions dont les colonnes sont les **attributs** et les lignes les **enregistrements**.

# Base de données : Vocabulaire

## Base de donnée

Une **base de donnée** est un ensemble de **tables**. On peut y extraire des informations à l'aide de **requêtes**.

## Table

Une **table** est un tableau à 2 dimensions dont les colonnes sont les **attributs** et les lignes les **enregistrements**.

Exemple : considérons une base de donnée astre contenant deux tables `planete` et `etoile`.

La table `planete` possède des attributs `nom`, `rayon`, `masse`...

Chaque enregistrement de `planete` correspond aux informations sur une planète.



## Base de données : Vocabulaire

| nom         | rayon (km) | masse (kg)         | etoile             |
|-------------|------------|--------------------|--------------------|
| 'Terre'     | 6400       | $6 \times 10^{24}$ | 'Soleil'           |
| 'Jupiter'   | 70000      | $2 \times 10^{27}$ | 'Soleil'           |
| 'Proxima b' | ?          | ?                  | 'Proxima Centauri' |
| ...         | ...        | ...                | ...                |

Table planete

| nom                | type          | duree_vie (année) | galaxie       |
|--------------------|---------------|-------------------|---------------|
| 'Soleil'           | 'Naine jaune' | $10^{10}$         | 'Voie lactée' |
| 'Proxima Centauri' | 'Naine rouge' | ?                 | 'Voie lactée' |
| ...                | ...           | ...               | ...           |

Table etoile

### Définition

Le **domaine** d'un attribut est l'ensemble des valeurs que peut prendre cet attribut.

## Définition

Le **domaine** d'un attribut est l'ensemble des valeurs que peut prendre cet attribut.

Dans la table `planete` :

- ① `nom` est un attribut ayant pour domaine l'ensemble des chaînes de caractères.
- ② `rayon` est un attribut ayant pour domaine  $\mathbb{N}$ .
- ③ ...

### Clé

Une **clé** d'une table est un ensemble minimal d'attributs permettant d'identifier de façon unique chaque enregistrement.

### Clé

Une **clé** d'une table est un ensemble minimal d'attributs permettant d'identifier de façon unique chaque enregistrement.

Parmi les clés possibles on en choisit une qu'on nomme **clé primaire**.

## Clé

Une **clé** d'une table est un ensemble minimal d'attributs permettant d'identifier de façon unique chaque enregistrement.

Parmi les clés possibles on en choisit une qu'on nomme **clé primaire**.

| nom                | type          | duree_vie (année) | galaxie       |
|--------------------|---------------|-------------------|---------------|
| 'Soleil'           | 'Naine jaune' | $10^{10}$         | 'Voie lactée' |
| 'Proxima Centauri' | 'Naine rouge' | ?                 | 'Voie lactée' |
| 'Kepler-22'        | 'Naine jaune' | ?                 | 'Voie lactée' |

Table etoile

Clés possibles ?

## Clé

Une **clé** d'une table est un ensemble minimal d'attributs permettant d'identifier de façon unique chaque enregistrement.

Parmi les clés possibles on en choisit une qu'on nomme **clé primaire**.

| nom                | type          | duree_vie (année) | galaxie       |
|--------------------|---------------|-------------------|---------------|
| 'Soleil'           | 'Naine jaune' | $10^{10}$         | 'Voie lactée' |
| 'Proxima Centauri' | 'Naine rouge' | ?                 | 'Voie lactée' |
| 'Kepler-22'        | 'Naine jaune' | ?                 | 'Voie lactée' |

Table etoile

Clés possibles ? Une seule possibilité : nom

## Base de données : Vocabulaire

| nom         | pays        | latitude | longitude |
|-------------|-------------|----------|-----------|
| 'Hanoï'     | 'Viêt Nam'  | 21°      | 104°      |
| 'Valence'   | 'France'    | 45°      | 5°        |
| 'Valence'   | 'Espagne'   | 39°      | 0°        |
| 'Quito'     | 'Equateur'  | 0°       | -78°      |
| 'Singapour' | 'Singapour' | 0°       | 104°      |
| 'Valence'   | 'France'    | 45°      | 1°        |

Table ville

### Question

Quelles sont les clés possibles ?



# Base de données : Vocabulaire

| nom         | pays        | latitude | longitude |
|-------------|-------------|----------|-----------|
| 'Hanoï'     | 'Viêt Nam'  | 21°      | 104°      |
| 'Valence'   | 'France'    | 45°      | 5°        |
| 'Valence'   | 'Espagne'   | 39°      | 0°        |
| 'Quito'     | 'Equateur'  | 0°       | -78°      |
| 'Singapour' | 'Singapour' | 0°       | 104°      |
| 'Valence'   | 'France'    | 45°      | 1°        |

Table ville

## Question

Quelles sont les clés possibles ?

- ① latitude, longitude
- ② nom, longitude
- ③ pays, longitude

On peut résumer la structure d'une table par son schéma :

## Schéma

Le schéma d'une table est la donnée de ses attributs, des domaines des attributs et de l'éventuelle clé primaire (soulignée), sous la forme :

table (attribut\_1 : type\_1, ..., attribut\_n : type\_n)

On peut résumer la structure d'une table par son schéma :

## Schéma

Le schéma d'une table est la donnée de ses attributs, des domaines des attributs et de l'éventuelle clé primaire (soulignée), sous la forme :

table (attribut\_1 : type\_1, ..., attribut\_n : type\_n)

Par exemple, le schéma de la table ville avec comme clé primaire (latitude, longitude) est :

ville (nom : chaîne de caractères, pays : chaîne de caractères,  
latitude : entier, longitude : entier)

On accède à des informations d'une base de donnée avec un **langage de requête**.

Contrairement à un langage de programmation :

- 1 On ne va pas utiliser de variable, boucle...
- 2 On se contente de demander ce que l'on veut obtenir, mais il n'y a pas besoin de dire comment l'obtenir : le moteur SQL se débrouille.

Pour trouver la somme des masses des planètes du système solaire :

❶ Langage de programmation :

---

```
Somme = 0
Pour toute planete p :
    Si p tourne autour du Soleil :
        Somme = Somme + p.masse
```

---

❷ Langage de requête :

---

```
Obtenir la somme des masse des planetes
qui tournent autour du Soleil
```

---

Quelques règles en SQL :

- ❶ Chaque requête est terminée par un point-virgule ;
- ❷ Pas d'indentation obligatoire comme en Python, mais il est conseillé de bien présenter son code
- ❸ Les commandes peuvent être écrites en majuscules ou minuscules
- ❹ Il est conseillé d'écrire les commandes SQL en majuscules et de donner des noms de tables et colonnes en minuscules

Les attributs peuvent être de type :

- **INT** : entier
- **CHAR**(*k*) : chaîne de caractères d'au plus *k* caractères  
Une chaîne de caractère doit être entourée d'apostrophes  
(**'exemple'**)
- **FLOAT** :  $\approx$  nombre à virgule
- **BOOLEAN** : booléen (en fait soit 0 soit 1)

# SQL : SELECT

Pour afficher des colonnes d'une table, on utilise :

```
SELECT colonne_1, ..., colonne_n FROM table;
```



# SQL : SELECT

Pour obtenir seulement les noms et prénoms des élèves dans une table `eleve(nom, prenom, classe, ecole)` :

```
SELECT nom, prenom FROM eleve;
```

# SQL : SELECT

Pour obtenir seulement les noms et prénoms des élèves dans une table `eleve(nom, prenom, classe, ecole)` :

```
SELECT nom, prenom FROM eleve;
```

On obtient :

| nom      | prenom |
|----------|--------|
| turing   | alan   |
| lovelace | ada    |
| erdős    | paul   |
| ...      | ...    |

# SQL : SELECT

Pour obtenir seulement les noms et prénoms des élèves dans une table `eleve(nom, prenom, classe, ecole)` :

```
SELECT nom, prenom FROM eleve;
```

On obtient :

| nom      | prenom |
|----------|--------|
| turing   | alan   |
| lovelace | ada    |
| erdős    | paul   |
| ...      | ...    |

Pour afficher la table entière, on peut utiliser `*` plutôt que donner le nom de chaque colonne :

```
SELECT * FROM eleve;
```

# SQL : SELECT

On peut faire des calculs dans les requêtes :

## Question

Que fait la requête suivante ?

---

```
SELECT nom, masse/((4/3)*3.14*(POW(rayon, 3)))  
FROM planete;
```

---

# SQL : SELECT

On peut faire des calculs dans les requêtes :

## Question

Que fait la requête suivante ?

```
SELECT nom, masse/((4/3)*3.14*(POW(rayon, 3)))  
FROM planete;
```

Elle affiche le nom et la densité de chaque planète.

## Exercice

masse est en kg et rayon en km.

Modifier la requête précédente pour afficher la densité en  $\text{g/cm}^3$ .

# SQL : DISTINCT

**DISTINCT** permet d'éviter d'avoir des doublons :

---

```
SELECT ecole FROM eleves;  
-- affiche plusieurs fois la même école  
-- (si plusieurs élèves l'ont intégré)
```

```
SELECT DISTINCT ecole FROM eleves;  
-- affiche une fois chaque école
```

---

# SQL : Fonctions d'agrégations

On peut utiliser des fonctions sur un attribut :

- **SUM(a)** : Somme l'attribut a sur tous les enregistrements.

Exemple : **SELECT SUM**(population) **FROM** world

# SQL : Fonctions d'agrégations

On peut utiliser des fonctions sur un attribut :

- **SUM(a)** : Somme l'attribut a sur tous les enregistrements.

Exemple : **SELECT SUM(population) FROM world**

- **MIN(a) / MAX(a)** : Minimum/maximum de a sur tous les enregistrements.

Exemple : **SELECT MAX(population) FROM world**



# SQL : Fonctions d'agrégations

On peut utiliser des fonctions sur un attribut :

- **SUM(a)** : Somme l'attribut a sur tous les enregistrements.  
Exemple : **SELECT SUM(population) FROM world**
- **MIN(a) / MAX(a)** : Minimum/maximum de a sur tous les enregistrements.  
Exemple : **SELECT MAX(population) FROM world**
- **COUNT(a)** : Compte le nombre de fois que a est différent de **null**.  
Souvent on compte le nombre total d'enregistrements avec **COUNT(\*)**.  
Exemple :  
**SELECT COUNT(\*) FROM eleve WHERE classe\_sup = 'MP2I'.**

# SQL : Fonctions d'agrégations

On peut utiliser des fonctions sur un attribut :

- **SUM(a)** : Somme l'attribut a sur tous les enregistrements.  
Exemple : **SELECT SUM(population) FROM world**
- **MIN(a) / MAX(a)** : Minimum/maximum de a sur tous les enregistrements.  
Exemple : **SELECT MAX(population) FROM world**
- **COUNT(a)** : Compte le nombre de fois que a est différent de **null**.  
Souvent on compte le nombre total d'enregistrements avec **COUNT(\*)**.  
Exemple :  
**SELECT COUNT(\*) FROM eleve WHERE classe\_sup = 'MP2I'**.
- **AVG(a)** : Moyenne de l'attribut a.  
Exemple :  
**SELECT AVG(note) FROM eleve WHERE classe\_sup = 'MP2I'**.

# SQL : Fonctions d'agrégations

On peut utiliser des fonctions sur un attribut :

- **SUM(a)** : Somme l'attribut a sur tous les enregistrements.  
Exemple : **SELECT SUM(population) FROM world**
- **MIN(a) / MAX(a)** : Minimum/maximum de a sur tous les enregistrements.  
Exemple : **SELECT MAX(population) FROM world**
- **COUNT(a)** : Compte le nombre de fois que a est différent de **null**.  
Souvent on compte le nombre total d'enregistrements avec **COUNT(\*)**.  
Exemple :  
**SELECT COUNT(\*) FROM eleve WHERE classe\_sup = 'MP2I'.**
- **AVG(a)** : Moyenne de l'attribut a.  
Exemple :  
**SELECT AVG(note) FROM eleve WHERE classe\_sup = 'MP2I'.**

Il est possible de renommer une colonne avec **AS** :

---

```
SELECT nom, masse/((4/3)*3.14*(POW(rayon, 3))) AS densite  
FROM planete;
```

---

Il est possible de renommer une colonne avec **AS** :

---

```
SELECT nom, masse/((4/3)*3.14*(POW(rayon, 3))) AS densite  
FROM planete;
```

---

Utile pour y faire référence ensuite :

---

```
SELECT nom, masse/((4/3)*3.14*(POW(rayon, 3))) AS densite  
FROM planete  
WHERE densite > 5.51
```

---

# SQL : WHERE

Il est possible de récupérer seulement les enregistrements vérifiant une condition avec **WHERE** :

```
SELECT colonne_1, ..., colonne_n FROM table  
WHERE condition;
```

# SQL : WHERE

Il est possible de récupérer seulement les enregistrements vérifiant une condition avec **WHERE** :

```
SELECT colonne_1, ..., colonne_n FROM table  
WHERE condition;
```

Dans condition on peut utiliser :

- ❶ = (et non pas ==)
- ❷ <, <=
- ❸ != (ou son équivalent <>)
- ❹ **AND**, **OR**
- ❺ **LIKE**

# SQL : WHERE

Pour afficher les noms des élèves qui ont été en MPSI2, on écrira :



# SQL : WHERE

Pour afficher les noms des élèves qui ont été en MPSI2, on écrira :

```
SELECT nom FROM eleve WHERE classe_sup = 'MPSI2';
```

Pour afficher les noms des élèves qui sont passés de MPSI à PSI :

# SQL : WHERE

Pour afficher les noms des élèves qui ont été en MPSI2, on écrira :

```
SELECT nom FROM eleve WHERE classe_sup = 'MPSI2';
```

Pour afficher les noms des élèves qui sont passés de MPSI à PSI :

---

```
SELECT nom FROM eleve  
WHERE classe_sup = 'PSI'  
AND (classe_pre = 'MPSI' OR classe_pre = 'MPSI2')
```

---

# SQL : WHERE

**LIKE** permet d'établir une condition sur la forme d'une chaîne de caractères d'un attribut :

attribut **LIKE** motif

# SQL : WHERE

**LIKE** permet d'établir une condition sur la forme d'une chaîne de caractères d'un attribut :

attribut **LIKE** motif

motif doit être une chaîne de caractères qui peut contenir :

- % : pour n'importe quelle chaîne de caractères
- \_ : pour n'importe quel (unique) caractère

# SQL : WHERE

## Question

Que fait la requête suivante ?

---

```
SELECT * from eleve  
WHERE ecole LIKE 'Centrale%'
```

---

# SQL : ORDER BY

**ORDER BY** permet de trier dans l'ordre croissant les enregistrements en fonction d'un attribut. On peut ajouter **DESC** pour trier dans l'ordre décroissant.

## Exemples :

- Noms d'élèves par ordre alphabétique :

---

```
SELECT nom  
FROM eleve  
ORDER BY nom;
```

---

- Planètes du système solaire de la plus lourde à la plus légère :

---

```
SELECT * FROM planete  
WHERE etoile = 'Soleil'  
ORDER BY masse DESC;
```

---

# SQL : LIMIT

**LIMIT**  $k$  permet de limiter le nombre d'enregistrements aux  $k$  premières valeurs. Il est souvent utilisé avec **ORDER BY**.

# SQL : LIMIT

**LIMIT**  $k$  permet de limiter le nombre d'enregistrements aux  $k$  premières valeurs. Il est souvent utilisé avec **ORDER BY**.

Exemple : Obtenir les 3 planètes les plus lourdes du système solaire.

---

```
SELECT * FROM planete
WHERE etoile = 'Soleil'
ORDER BY masse DESC
LIMIT 3;
```

---



Toutes les commandes optionnelles de SELECT doivent être **écrites dans cet ordre** :

---

```
SELECT colonne1, colonne2, ...  
FROM table  
WHERE condition  
ORDER BY colonne  
LIMIT k
```

---

# Exercices

planete (nom, masse, rayon)

## Question

Comment obtenir, dans la table `planete`, la deuxième planète la plus dense connue?

```
planete (nom, masse, rayon)
```

## Question

Comment obtenir, dans la table `planete`, la deuxième planète la plus dense connue?

```
eleve (nom, annee_entree, ecole, classe_sup,  
       classe_spe, classe_spe2)
```

## Question

Comment obtenir, dans la table `eleve`, les noms des 3 derniers élèves entrés dans une ENS en MP\*?

# NULL

## NULL

**NULL** est un mot-clé signifiant « valeur manquante ».

# NULL

## NULL

**NULL** est un mot-clé signifiant « valeur manquante ».

Pour savoir si un attribut est **NULL** :

**WHERE ... IS NULL**

(et non pas **WHERE ... = NULL**)

# NULL

## NULL

**NULL** est un mot-clé signifiant « valeur manquante ».

Pour savoir si un attribut est **NULL** :

**WHERE ... IS NULL**

(et non pas **WHERE ... = NULL**)

Pour savoir si un attribut n'est pas **NULL** :

**WHERE ... IS NOT NULL**

(et non pas **WHERE ... <> NULL**)