Éliminer les doublons avec DISTINCT

Dans quelles communes habitent les élèves ? DISTINCT affiche chaque valeur une seule fois. C'est surtout un outil de contrôle : une valeur de trop dans la liste, et une faute de frappe saute aux yeux.

4TTR 5TTR 6TTR Découverte

Dans quelles communes habitent les élèves de l'école ? La réponse tient en quelques noms. Pourtant, la colonne `commune` en contient douze, un par élève, avec beaucoup de répétitions. Il faut demander à la base de ne garder chaque commune qu'une fois.

> 💡 Télécharge `mon-ecole-eleves.db` dans la sidebar, ouvre-la dans DB Browser for SQLite, puis va dans l'onglet **Exécuter le SQL**. Tape chaque requête dans la zone du haut et exécute-la avec ▶ ou Ctrl+Entrée.

🎯 Objectifs

À la fin de cet article, tu seras capable de :

1. afficher les valeurs différentes d'une colonne avec `DISTINCT` ;
2. utiliser `DISTINCT` pour repérer des fautes de frappe dans les données ;
3. prévoir le résultat de `DISTINCT` sur plusieurs colonnes ;
4. combiner `DISTINCT` avec `WHERE` et `ORDER BY`.

Une colonne pleine de répétitions

La base contient une table `eleve`, avec une ligne par élève. Pour la lire, on écrit une **requête** : `SELECT` (*sélectionne*) donne les colonnes à afficher, `FROM` (*depuis*) donne la table où les chercher.

Affiche la colonne `commune` :

```
SELECT commune
FROM eleve;
```

```
commune
---------
Jodoigne
Wavre
Perwez
Jodoigne
Hannut
Jodoigne
Wavre
Perwez
Jodoigne
Hannut
Ramillies
jodoigne
```

Douze lignes, une par élève. Pour savoir combien de communes différentes il y a, il faudrait barrer les répétitions à la main.

Supprimer les répétitions avec DISTINCT

Ajoute le mot `DISTINCT` juste après `SELECT` :

```sql
SELECT DISTINCT commune
FROM eleve;
```

```
commune
---------
Jodoigne
Wavre
Perwez
Hannut
Ramillies
jodoigne
```

`DISTINCT` veut dire *distinct*, c'est-à-dire *différent*. SQLite construit le résultat, puis supprime les lignes qui sont la copie exacte d'une ligne déjà gardée.

> 📖 **NOUVELLE NOTION : DISTINCT**
>
> `SELECT DISTINCT` affiche les lignes différentes du résultat, **sans répétition** : chaque valeur n'apparaît qu'une fois. Deux lignes identiques sont appelées des **doublons** ; `DISTINCT` n'en garde qu'un exemplaire. La table, elle, n'est pas modifiée.
>
> En anglais : GB ***distinct***, et GB ***duplicate*** pour un doublon.

`DISTINCT` se place toujours **juste après** `SELECT`, avant les colonnes. Écrit ailleurs, il provoque une erreur :

```sql
SELECT commune DISTINCT
FROM eleve;
```

```
near "DISTINCT": syntax error
```

Un outil pour contrôler la qualité des données

Relis le résultat de `SELECT DISTINCT commune`. Il contient **six** lignes. Or les élèves habitent cinq communes : Jodoigne, Wavre, Perwez, Hannut et Ramillies. D'où vient la sixième ?

Ajoute un tri pour y voir plus clair. `ORDER BY` (*ordonner par*) range les lignes par ordre alphabétique :

```sql
SELECT DISTINCT commune
FROM eleve
ORDER BY commune;
```

```
commune
---------
Hannut
Jodoigne
Perwez
Ramillies
Wavre
jodoigne
```

La faute saute aux yeux : `jodoigne`, écrit en minuscules, est une faute de frappe sur la ligne d'un élève. SQLite le range après Wavre, parce qu'il trie les majuscules avant les minuscules.

Pour SQLite, `Jodoigne` et `jodoigne` sont deux textes différents, donc deux valeurs distinctes. Cette faute a une conséquence concrète : une requête `WHERE commune = 'Jodoigne'` ne trouve que quatre élèves au lieu de cinq. L'élève mal encodée disparaît de toutes les listes de Jodoigne.

> ⚠️ La base a accepté cette faute sans aucun message d'erreur. Rien ne lui dit que `jodoigne` n'est pas une commune valable. Une faute de ce genre ne se voit que si on la cherche.

> 🧠 `SELECT DISTINCT colonne ... ORDER BY colonne` est un réflexe de contrôle. Sur une colonne qui ne devrait contenir qu'un petit nombre de valeurs, toute valeur de trop est suspecte : faute de frappe, majuscule oubliée, espace en trop.

Le même contrôle fonctionne sur les nombres. Voici les différents nombres d'absences :

```sql
SELECT DISTINCT absences
FROM eleve
ORDER BY absences;
```

```
absences
--------
0
1
2
3
4
5
8
12
```

Ici, rien de suspect : toutes les valeurs sont plausibles. Un nombre négatif, ou un 950, aurait mérité une vérification.

> ℹ️ Pour corriger la faute, il faut modifier la donnée elle-même avec une requête `UPDATE`, qui n'est pas le sujet de cet article. `DISTINCT` sert à la **trouver**, pas à la réparer.

DISTINCT avec WHERE

`DISTINCT` se combine avec `WHERE` (*où*), qui ne garde que les lignes qui remplissent une condition. SQLite filtre d'abord les lignes, puis supprime les doublons. Voici les communes où habitent les élèves nés avant 2010 :

```sql
SELECT DISTINCT commune
FROM eleve
WHERE date_naissance < '2010-01-01'
ORDER BY commune;
```

```
commune
---------
Jodoigne
Perwez
Ramillies
```

Quatre élèves sont nés avant 2010, mais deux habitent Jodoigne : il reste trois communes.

DISTINCT sur plusieurs colonnes

Quand `DISTINCT` est suivi de plusieurs colonnes, il ne regarde pas chaque colonne séparément. Il compare des **lignes entières** : deux lignes sont des doublons seulement si **toutes** leurs colonnes sont identiques.

```sql
SELECT DISTINCT commune, absences
FROM eleve
ORDER BY commune, absences;
```

```
commune   | absences
----------+---------
Hannut    | 4
Hannut    | 8
Jodoigne  | 0
Jodoigne  | 1
Jodoigne  | 2
Jodoigne  | 12
Perwez    | 0
Perwez    | 5
Ramillies | 1
Wavre     | 0
Wavre     | 3
jodoigne  | 0
```

Douze lignes : aucune n'a été supprimée. `Jodoigne` apparaît quatre fois, mais chaque fois avec un nombre d'absences différent. Aucune paire commune-absences ne se répète, donc il n'y a aucun doublon à supprimer.

⚠️ `DISTINCT` s'applique à **toutes** les colonnes du `SELECT`, jamais à une seule. Des parenthèses n'y changent rien : `SELECT DISTINCT(commune), nom` compare toujours les paires commune-nom, et renvoie les douze lignes.

ℹ️ Pour `DISTINCT`, les cases vides sont considérées comme identiques entre elles. `SELECT DISTINCT email FROM eleve` renvoie dix lignes : les neuf adresses, plus une seule ligne `NULL` pour les trois élèves sans adresse.

🧪 Exercices

Pour chaque exercice, écris la requête, exécute-la, puis vérifie que le résultat répond vraiment à la question.

Exercice 1 — Lister les valeurs ★☆☆☆☆

Écris une requête qui affiche, sans répétition et dans l'ordre croissant :

1. les différents nombres d'absences ;
2. les communes des élèves qui ont plus de 2 absences ;
3. les communes des élèves nés en 2010 ;

4. les années de naissance… en réfléchissant d'abord : est-ce possible avec ce que tu connais ? Pourquoi `SELECT DISTINCT date_naissance` ne répond-il pas à la question ?

Exercice 2 — Diagnostiquer la colonne commune ★★☆☆☆

1. Exécute `SELECT DISTINCT commune FROM eleve ORDER BY commune;`. Combien de valeurs obtiens-tu ? Combien de communes réelles ?
2. Écris une requête qui affiche le prénom et le nom de l'élève dont la commune est mal encodée.
3. Explique pourquoi cette valeur arrive en fin de liste.
4. La direction envoie un courrier aux familles de Jodoigne avec `WHERE commune = 'Jodoigne'`. Que se passe-t-il pour cette famille ?

Exercice 3 — Prédire le résultat ★★☆☆☆

Sans exécuter, écris combien de lignes renvoie chaque requête. Exécute ensuite pour vérifier.

```sql
-- a)
SELECT DISTINCT commune FROM eleve WHERE commune LIKE 'jodoigne';

-- b)
SELECT DISTINCT commune, absences FROM eleve;

-- c)
SELECT DISTINCT absences FROM eleve WHERE absences = 0;

-- d)
SELECT DISTINCT email FROM eleve;
```

Pour la requête a), sache que `LIKE` ne fait pas la différence entre majuscules et minuscules pour les lettres sans accent.

Exercice 4 — Réparer des requêtes ★★★☆☆

Chacune de ces requêtes provoque une erreur ou ne répond pas à la question. Pour chacune : exécute-la, observe, explique ce qui ne va pas, puis corrige-la.

```sql
-- a) La liste des communes, sans répétition
SELECT commune DISTINCT FROM eleve;

-- b) La liste des communes, sans répétition, par ordre alphabétique
SELECT DISTINCT commune FROM eleve ORDER BY;

-- c) La liste des communes, sans répétition
SELECT DISTINCT(commune), nom FROM eleve;
```

✍️ À retenir

- `SELECT DISTINCT` (*distinct, différent*) affiche chaque ligne du résultat une seule fois.
- `DISTINCT` se place juste après `SELECT`, avant les colonnes.
- Sur plusieurs colonnes, ce sont les lignes entières qui doivent être identiques pour être fusionnées.
- `SELECT DISTINCT colonne ... ORDER BY colonne` est un outil de contrôle : une valeur de trop signale souvent une faute de frappe.
- Pour SQLite, `Jodoigne` et `jodoigne` sont deux valeurs distinctes.
- `DISTINCT` trouve les fautes, il ne les corrige pas.

Suite

`DISTINCT` montre les communes, mais pas combien d'élèves habitent chacune. Pour compter, additionner, faire des moyennes et obtenir un résultat par commune, passe à Compter et regrouper.