Chercher un motif avec LIKE

Un nom qui commence par H, une adresse qui se termine par @ecole.be, une commune écrite avec ou sans majuscule : LIKE cherche un motif plutôt qu'une valeur exacte.

4TTR 5TTR 6TTR Découverte

Tu cherches un élève dont le nom commence par D, mais tu as oublié la suite. Tu veux vérifier que toutes les adresses email se terminent par `@ecole.be`. Dans les deux cas, tu ne connais pas la valeur exacte, seulement une partie. Le signe `=` ne peut pas t'aider.

> 💡 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. chercher un motif dans du texte avec `LIKE` ;
2. utiliser les jokers `%` et `_` ;
3. exclure un motif avec `NOT LIKE` ;
4. prévoir quand `LIKE` tient compte des majuscules et des accents ;
5. utiliser `LIKE` pour retrouver une valeur mal encodée.

Pourquoi le signe égal ne suffit pas

La base contient une table `eleve`, avec une ligne par élève. Pour la lire, on écrit une **requête** : `SELECT` (*sélectionne*) donne les colonnes, `FROM` (*depuis*) donne la table, et `WHERE` (*où*) ne garde que les lignes qui remplissent une condition.

Première idée pour trouver les noms qui commencent par D :

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

```
3 | WHERE nom = 'D%';
```

```
1 | prenom | nom
2 | -------+----
```

Aucune ligne. Le signe `=` compare exactement : il cherche un élève dont le nom est littéralement `D%`, et il n'y en a pas.

Chercher un motif avec LIKE

Remplace `=` par `LIKE` :

```
1 | SELECT prenom, nom
2 | FROM eleve
3 | WHERE nom LIKE 'D%';
```

```
1 | prenom | nom
2 | -------+--------
3 | Léa    | Delvaux
```

`LIKE` veut dire *comme*, au sens de *qui ressemble à*. La condition se lit : « où le nom ressemble à D suivi de n'importe quoi ». Dans `'D%'`, le symbole `%` remplace n'importe quelle suite de caractères.

> 📖 **NOUVELLE NOTION : LE MOTIF**
>
> Un **motif** est un modèle de texte qui décrit une famille de valeurs, par exemple « commence par D ». `LIKE` (*comme*) garde les lignes dont la valeur correspond au motif. Un motif s'écrit entre apostrophes, comme n'importe quel texte.
>
> En anglais : GB ***pattern***.

Les deux jokers

Le `%` n'est pas le seul symbole spécial. `LIKE` en connaît deux.

Joker	Remplace	Exemple	Correspond à
`%`	zéro, un ou plusieurs caractères	`'D%'`	`D`, `Delvaux`, `Dubois`
`_`	exactement un caractère	`'G_rard'`	`Gérard`, `Girard`

📖 NOUVELLE NOTION : LE JOKER

Un **joker** est un caractère qui, dans un motif, remplace d'autres caractères. Avec `LIKE`, `%` remplace n'importe quelle suite de caractères, même vide, et `_` remplace exactement un caractère. Comme au jeu de cartes, le joker peut prendre la place de n'importe quelle carte.

En anglais : GB ***wildcard***.

La position du `%` change tout. Voici les noms qui **se terminent** par t :

```sql
SELECT prenom, nom
FROM eleve
WHERE nom LIKE '%t';
```

```
prenom | nom
-------+----------
Hugo   | Englebert
Manon  | Hubert
Sarah  | Lambert
```

Avec un `%` de chaque côté, on cherche les noms qui **contiennent** `er`, n'importe où :

```sql
SELECT prenom, nom
FROM eleve
WHERE nom LIKE '%er%';
```

```
prenom | nom
-------+----------
Noah   | Charlier
Hugo   | Englebert
Manon  | Hubert
Arthur | Kevers
Sarah  | Lambert
```

Le `_` compte les caractères un par un. Quatre `_` à la suite trouvent les prénoms de quatre lettres exactement :

```sql
SELECT prenom, nom
FROM eleve
WHERE prenom LIKE '____';
```

```
prenom | nom
-------+----------
Emma   | Bastin
Noah   | Charlier
Hugo   | Englebert
```

Les deux jokers peuvent se combiner. `'_a%'` veut dire : un caractère quelconque, puis un `a`, puis n'importe quoi. C'est-à-dire les prénoms dont la deuxième lettre est un a.

```sql
SELECT prenom, nom
FROM eleve
WHERE prenom LIKE '_a%';
```

```
prenom  | nom
--------+--------
Nathan  | Gérard
Manon   | Hubert
Camille | Jacques
Sarah   | Lambert
```

⚠️ L'étoile `*` n'est pas un joker en SQL, même si tu l'as peut-être utilisée pour chercher des fichiers dans Windows. `WHERE nom LIKE 'D*'` cherche un nom qui est littéralement `D*`, et ne renvoie rien.

⚠️ Un `LIKE` sans joker se comporte presque comme `=`. `WHERE prenom LIKE 'Emm'` ne trouve pas Emma : le motif ne dit pas qu'il peut y avoir quelque chose après.

Chercher dans les adresses email

Les motifs sont très utiles pour contrôler des adresses email. Vérifie qu'elles appartiennent toutes au domaine de l'école :

```sql
SELECT prenom, nom, email
FROM eleve
WHERE email LIKE '%@ecole.be';
```

```
prenom | nom       | email
-------+-----------+-----------------------
Lucas  | Adam      | lucas.adam@ecole.be
Emma   | Bastin    | emma.bastin@ecole.be
Léa    | Delvaux   | lea.delvaux@ecole.be
Hugo   | Englebert | hugo.englebert@ecole.be
Nathan | Gérard    | nathan.gerard@ecole.be
Manon  | Hubert    | manon.hubert@ecole.be
Louis  | Istas     | louis.istas@ecole.be
Arthur | Kevers    | arthur.kevers@ecole.be
Sarah  | Lambert   | sarah.lambert@ecole.be
```

Neuf lignes, pas douze. Noah Charlier, Chloé Fontaine et Camille Jacques n'ont pas d'adresse email : leur case est vide, marquée `NULL` dans la base. Une case vide ne ressemble à aucun motif, donc la condition n'est jamais vraie pour eux.

Les adresses sont construites sur le modèle `prenom.nom@ecole.be`. Un motif peut donc retrouver les élèves dont le prénom commence par l :

```sql
SELECT prenom, nom, email
FROM eleve
WHERE email LIKE 'l%';
```

```
prenom | nom     | email
-------+---------+--------------------
Lucas  | Adam    | lucas.adam@ecole.be
Léa    | Delvaux | lea.delvaux@ecole.be
Louis  | Istas   | louis.istas@ecole.be
```

Exclure un motif avec NOT LIKE

`NOT LIKE` (*pas comme*) garde les lignes qui **ne** correspondent **pas** au motif. Voici les noms qui ne contiennent pas la lettre e :

```sql
SELECT prenom, nom
FROM eleve
WHERE nom NOT LIKE '%e%';
```

```
prenom | nom
-------+-------
Lucas  | Adam
Emma   | Bastin
Nathan | Gérard
Louis  | Istas
```

Gérard est dans la liste, alors qu'on y lit un é. C'est normal : pour `LIKE`, `é` et `e` sont deux caractères différents. La section suivante revient sur ce point.

Majuscules, minuscules et accents

Dans cette base, une faute s'est glissée : la commune de Sarah Lambert est écrite `jodoigne`, en minuscules, alors que les quatre autres élèves de la commune ont `Jodoigne`. Avec `WHERE commune = 'Jodoigne'`, on n'obtient que quatre élèves.

Essaie avec `LIKE`, sans aucun joker :

```sql
SELECT prenom, nom, commune
FROM eleve
WHERE commune LIKE 'jodoigne';
```

```
prenom | nom      | commune
-------+----------+---------
Lucas  | Adam     | Jodoigne
Léa    | Delvaux  | Jodoigne
Chloé  | Fontaine | Jodoigne
Louis  | Istas    | Jodoigne
Sarah  | Lambert  | jodoigne
```

Les cinq élèves sont là. Dans SQLite, `LIKE` **ne fait pas la différence entre majuscules et minuscules** pour les lettres sans accent. `LIKE 'JODOIGNE'` donnerait exactement le même résultat.

Avec les lettres accentuées, c'est une autre histoire :

```sql
SELECT prenom, nom
FROM eleve
WHERE nom LIKE 'GÉRARD';
```

```
prenom | nom
-------+----
```

Aucune ligne. SQLite sait que `G` et `g` sont la même lettre, mais pas que `É` et `é` le sont. Écrit `'gérard'`, en minuscules avec le même accent, le motif trouve Nathan Gérard. Écrit `'Gerard'`, sans accent, il ne trouve rien non plus.

Condition	Trouve Gérard ?	Pourquoi
`nom LIKE 'gérard'`	oui	seules les lettres sans accent changent de casse
`nom LIKE 'GÉRARD'`	non	`É` et `é` sont vus comme différents
`nom LIKE 'Gerard'`	non	`e` et `é` sont deux caractères différents
`nom LIKE 'G_rard'`	oui	le `_` remplace le é

🧠 Dans SQLite, `LIKE` ignore la différence entre majuscules et minuscules, mais seulement pour les 26 lettres sans accent. Un accent, lui, compte toujours.

ℹ️ Ce comportement dépend du logiciel. Dans PostgreSQL, `LIKE` fait la différence entre majuscules et minuscules. Dans MySQL, cela dépend de la configuration de la table. Vérifie toujours avant de compter dessus.

🧪 Exercices

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

Exercice 1 — Utiliser les jokers ★☆☆☆☆

Écris une requête qui affiche le prénom et le nom :

1. des élèves dont le nom commence par H ;
2. des élèves dont le prénom se termine par a ;
3. des élèves dont le prénom contient `ou` ;
4. des élèves dont le nom compte exactement six lettres ;
5. des élèves dont le nom ne commence ni par A ni par E.

Exercice 2 — Prédire le résultat ★★☆☆☆

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

```sql
-- a)
SELECT prenom FROM eleve WHERE prenom LIKE 'L%';

-- b)
SELECT nom FROM eleve WHERE nom LIKE '_a%';

-- c)
SELECT prenom FROM eleve WHERE prenom LIKE 'léa';

-- d)
SELECT prenom FROM eleve WHERE prenom LIKE 'LÉA';

-- e)
SELECT email FROM eleve WHERE email NOT LIKE '%@ecole.be';
```

Pour la requête e), explique le résultat.

Exercice 3 — Contrôler les emails ★★☆☆☆

1. Affiche les élèves dont l'adresse email se termine par `@ecole.be`. Combien sont-ils ? Pourquoi pas douze ?
2. Affiche les adresses dont la partie avant le point compte exactement quatre caractères.
3. Affiche le prénom et l'email des élèves dont l'adresse contient `gerard`. Le nom de famille de l'élève contient-il un accent ? Et son adresse ?

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) Les noms qui commencent par H
SELECT nom FROM eleve WHERE nom LIKE 'H*';

-- b) Les noms qui commencent par H
SELECT nom FROM eleve WHERE nom = 'H%';

-- c) Les noms qui commencent par H
SELECT nom FROM eleve WHERE nom LIKE H%;

-- d) Nathan Gérard
SELECT prenom, nom FROM eleve WHERE nom LIKE 'Gerard';

-- e) Les prénoms de quatre lettres
SELECT prenom FROM eleve WHERE prenom LIKE '___';
```

Exercice 5 — La commune mal encodée ★★☆☆☆

1. Compte les lignes renvoyées par `WHERE commune = 'Jodoigne'`, puis par `WHERE commune LIKE 'jodoigne'`.
2. Explique la différence en deux phrases.
3. Écris une requête qui affiche **uniquement** la ligne mal encodée.

✍️ À retenir

- `LIKE` (*comme*) compare une valeur à un motif, écrit entre apostrophes.
- `%` remplace zéro, un ou plusieurs caractères ; `_` remplace exactement un caractère.
- `'D%'` commence par D, `'%t'` se termine par t, `'%er%'` contient er.
- `NOT LIKE` garde les lignes qui ne correspondent pas au motif.
- Dans SQLite, `LIKE` ignore la casse pour les lettres sans accent, mais pas pour les lettres accentuées.
- Une case vide (`NULL`) ne correspond à aucun motif, ni avec `LIKE` ni avec `NOT LIKE`.

Suite

Les trois élèves sans adresse email échappent à `LIKE` comme à `NOT LIKE`. Pour comprendre ce que contiennent ces cases vides et comment les retrouver, passe à Les valeurs absentes : NULL.