

Référence rapide SQL

SELECT, JOIN, sous-requêtes, index, transactions

SELECT

```
SELECT * FROM users;
SELECT name, email FROM users;
SELECT DISTINCT city FROM users;
SELECT name AS full_name FROM users;
```

WHERE

Opérateurs de comparaison

= <> (!=)	Égal / différent
< > <= >=	Opérateurs de comparaison
AND OR NOT	Opérateurs logiques
IS NULL / IS NOT NULL	Vérifications de valeur nulle

Correspondance de motifs

```
SELECT * FROM users WHERE name LIKE 'A%';
-- % = any chars, _ = single char
SELECT * FROM users WHERE age IN (20, 25, 30);
SELECT * FROM users WHERE age BETWEEN 18 AND 30;
```

JOIN

Types de jointure

INNER JOIN	Lignes correspondantes dans les deux tables
LEFT JOIN	Toutes les lignes gauches + correspondances droites
RIGHT JOIN	Toutes les lignes droites + correspondances gauches
FULL OUTER JOIN	Toutes les lignes des deux tables
CROSS JOIN	Produit cartésien des deux tables

Syntaxe de jointure

```
SELECT u.name, o.total
FROM users u
INNER JOIN orders o ON u.id = o.user_id;

SELECT u.name, o.total
FROM users u
LEFT JOIN orders o ON u.id = o.user_id;
```

INSERT / UPDATE / DELETE

Insertion

```
INSERT INTO users (name, email)
VALUES ('Alice', 'alice@example.com');

INSERT INTO users (name, email) VALUES
('Bob', 'bob@ex.com'),
('Carol', 'carol@ex.com');
```

Mise à jour

```
UPDATE users SET email = 'new@ex.com'
WHERE id = 1;
```

Suppression

```
DELETE FROM users WHERE id = 1;
DELETE FROM users; -- delete all rows
```

CREATE TABLE

Syntaxe

```
CREATE TABLE users (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  name TEXT NOT NULL,
  email TEXT UNIQUE,
  age INTEGER DEFAULT 0,
  score REAL
);
```

Types de données courants

INTEGER	Nombres entiers
REAL	Nombres à virgule flottante
TEXT	Données texte / chaîne
BLOB	Données binaires
BOOLEAN	TRUE / FALSE (stocké en 0/1)
DATE / DATETIME	Valeurs de date et horodatage

Contraintes

PRIMARY KEY	Identifiant unique de ligne
NOT NULL	Valeur obligatoire
UNIQUE	Pas de valeurs dupliquées
DEFAULT val	Valeur par défaut si omise
CHECK (expr)	Règle de validation personnalisée
FOREIGN KEY	Référence à une autre table

Fonctions d'agrégation

COUNT(*)	Nombre de lignes
COUNT(col)	Valeurs non nulles dans la colonne
SUM(col)	Somme d'une colonne numérique
AVG(col)	Moyenne d'une colonne numérique
MIN(col)	Valeur minimale
MAX(col)	Valeur maximale

Exemple

```
SELECT COUNT(*) AS total,
       AVG(age) AS avg_age,
       MAX(score) AS top_score
FROM users;
```

GROUP BY / HAVING

```
SELECT city, COUNT(*) AS num_users
FROM users
GROUP BY city;

SELECT city, AVG(age) AS avg_age
FROM users
GROUP BY city
HAVING AVG(age) > 25;
```

WHERE filtre les lignes avant le groupement ; HAVING filtre les groupes après l'agrégation

ORDER BY / LIMIT

```
SELECT * FROM users ORDER BY name ASC;
SELECT * FROM users ORDER BY age DESC;
SELECT * FROM users
ORDER BY city, name
LIMIT 10 OFFSET 20; -- skip 20, take 10
```

Sous-requêtes

Dans la clause WHERE

```
SELECT name FROM users
WHERE id IN (
  SELECT user_id FROM orders
  WHERE total > 100
);
```

En tant que table dérivée

```
SELECT city, avg_age FROM (
  SELECT city, AVG(age) AS avg_age
  FROM users GROUP BY city
) WHERE avg_age > 30;
```

Index

```
CREATE INDEX idx_name ON users(name);
CREATE UNIQUE INDEX idx_email
ON users(email);
DROP INDEX idx_name;
```

Quand indexer

Colonnes dans WHERE	Accélérer le filtrage
Colonnes dans JOIN ON	Accélérer les recherches de jointure
Colonnes dans ORDER BY	Accélérer le tri
Colonnes à haute cardinalité	Bénéfice maximal avec beaucoup de valeurs uniques