SQL Partie 1

Une introduction pratique aux requêtes fondamentales pour développeurs et ingénieurs data

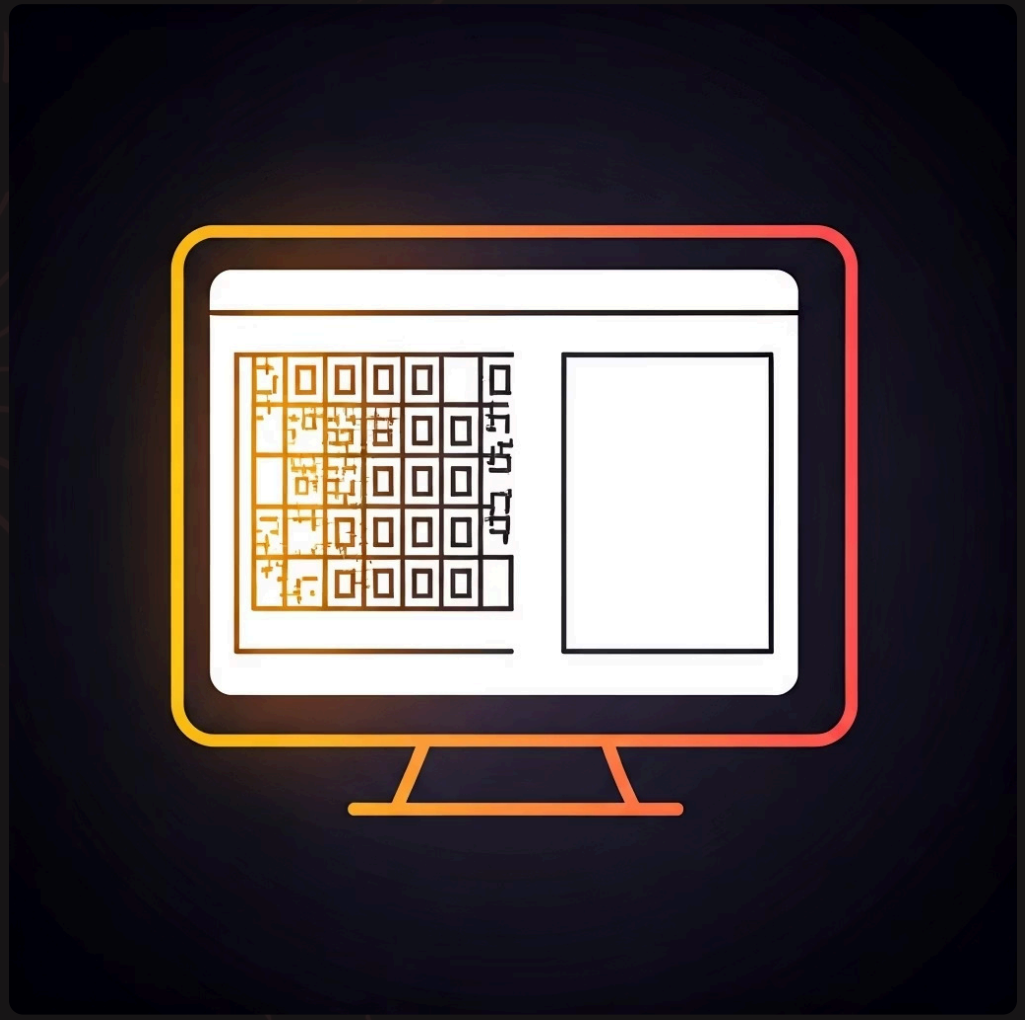
L'Art de la Sélection : SELECT

Votre Premier Contact

Lire une table entière, c'est le point de départ de toute exploration SQL. La commande SELECT * vous donne une vue complète de vos données.

```
SELECT * FROM flights;
```

Simple, direct, efficace pour découvrir la structure de vos données.



Sélection Ciblée et Alias

Principe Clé

Ne récupérez que ce dont vous avez besoin. Renommez pour plus de clarté.

Syntaxe

```
SELECT colonne AS alias
FROM table;
```

Exemple Pratique

```
SELECT id,
       first_name AS prénom,
       last_name AS nom
FROM employee;
```

Les alias rendent vos résultats plus lisibles et professionnels, surtout dans les rapports.

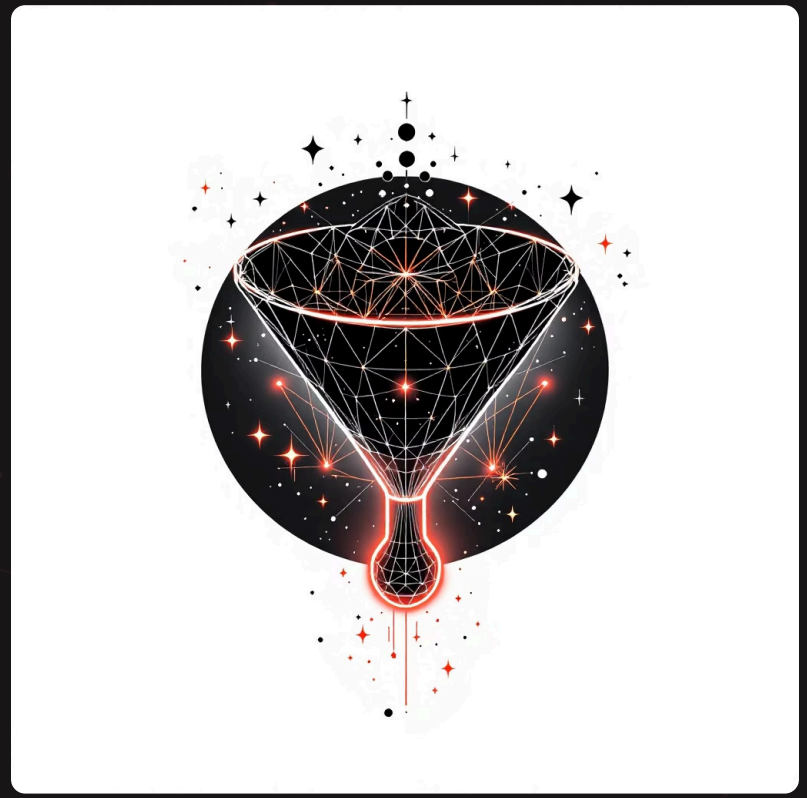
Filtrage avec WHERE

Filtrer Efficacement

WHERE est votre outil de précision. Il transforme une requête générale en recherche ciblée.

```
SELECT first_name, salary
FROM employee
WHERE salary > 50000;
```

Trouvez exactement les enregistrements qui correspondent à vos critères métier.



Tri et Limitation

1

ORDER BY

Organisez vos résultats par ordre croissant (ASC) ou décroissant (DESC)

2

LIMIT

Contrôlez le nombre de résultats retournés pour optimiser les performances

```
SELECT first_name, salary
FROM employee
ORDER BY salary DESC
LIMIT 3;
```

Parfait pour identifier rapidement les top performers ou les premiers enregistrements.

Fonctions d'Agrégation

COUNT(*)

Comptez le nombre total d'enregistrements

```
SELECT COUNT(*) FROM sales;
```

AVG() & GROUP BY

Calculez des moyennes par groupes

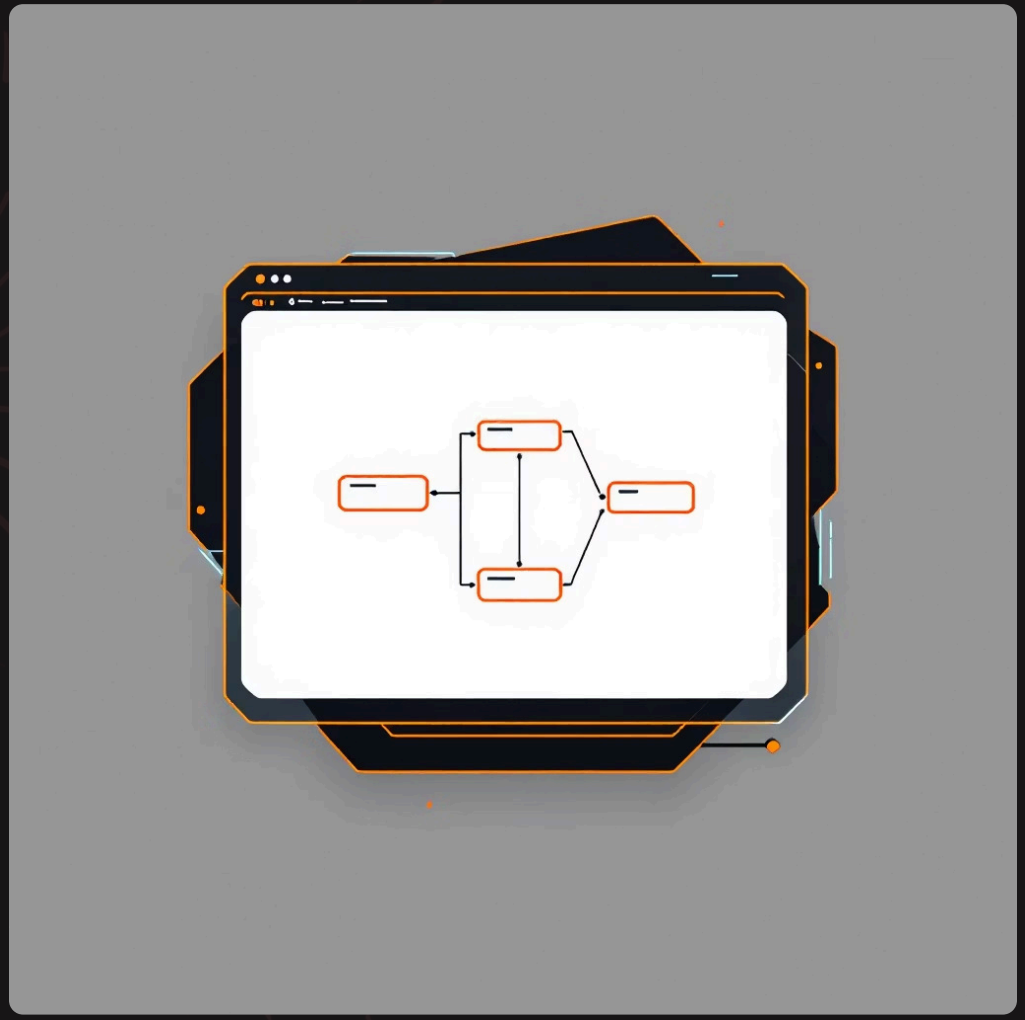
```
SELECT department_id, AVG(salary)
FROM employee
GROUP BY department_id;
```

INNER JOIN : Connexions Essentielles

Relier les Tables

INNER JOIN combine les données de deux tables quand une correspondance existe. C'est le fondement du travail avec des données relationnelles.

```
SELECT e.first_name, d.name AS department
FROM employee e
INNER JOIN department d
    ON e.department_id = d.id;
```



Types de Jointures

LEFT JOIN

Tous les enregistrements de gauche + correspondances de droite

```
SELECT e.first_name, d.name
FROM employee e
LEFT JOIN department d
    ON e.department_id = d.id;
```

FULL JOIN

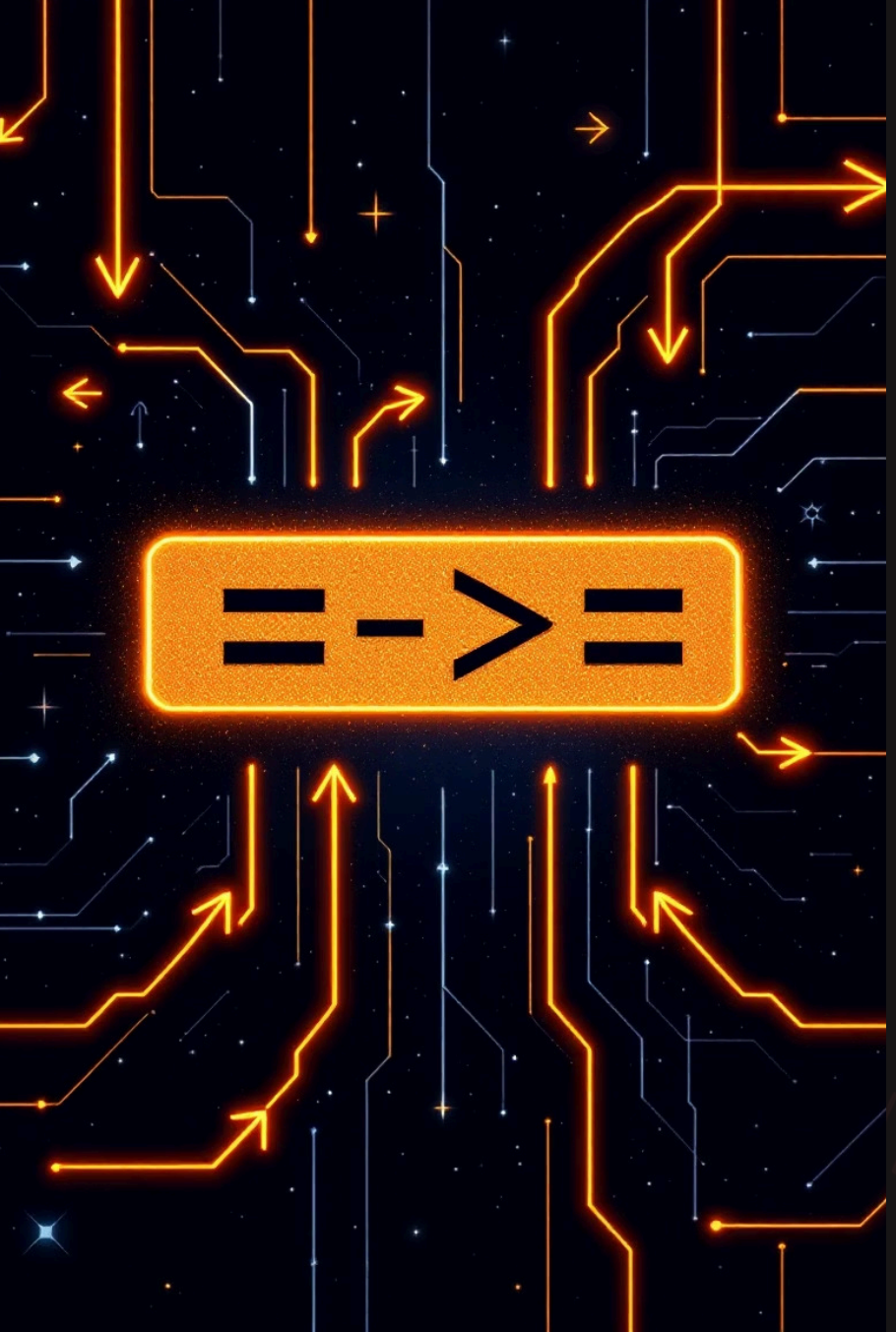
Tous les enregistrements des deux tables, même sans correspondance

```
SELECT e.first_name, d.name
FROM employee e
FULL JOIN department d
    ON e.department_id = d.id;
```

CROSS JOIN

Produit cartésien : chaque ligne de gauche avec chaque ligne de droite

```
SELECT e.first_name, d.name
FROM employee e
CROSS JOIN department d;
```



Conversion de Types

CAST vs ::

PostgreSQL offre deux syntaxes pour convertir les types de données. Les deux méthodes sont équivalentes.

Syntaxe Standard SQL :

```
SELECT CAST(salary
AS text)
FROM employee;
```

Syntaxe PostgreSQL :

```
SELECT salary::text
FROM employee;
```

La syntaxe :: est plus concise et couramment utilisée par les développeurs PostgreSQL.

Logique Conditionnelle et Gestion des NULL

CASE WHEN

Créez des catégories dynamiques dans vos résultats

```
SELECT first_name, salary,
CASE
  WHEN salary > 50000 THEN 'Haut'
  WHEN salary > 30000 THEN 'Moyen'
  ELSE 'Bas'
END AS niveau_salaire
FROM employee;
```

COALESCE

Remplacez élégamment les valeurs NULL

```
SELECT first_name,
COALESCE(email, 'Non renseigné') AS email
FROM employee;
```

Ces deux fonctions transforment vos données brutes en informations exploitables et professionnelles.