TP 1: SQL SELECT Statements

Objectifs :

Se familiariser avec les commandes de base du langage SQL.

Modalités pédagogique:

Travail individuel

Durée : Un jour

Matériel :

Postgre SQL installé sur votre machine (Mac, linux ou Windows)

Lien : https://www.postgresql.org/download/

Dataset :

Prenez le temps et consultez le dataset suivant avant de faire les exercices, **il est important d'analyser un dataset avant de faire des manipulations dessus** 😉 :

https://www.kaggle.com/datasets/ravindrasinghrana/employeedataset

Objectifs pédagogiques

Théorie/Entrainement :

SELECT

1- Sélectionner une table

Syntaxe :

```
SELECT * FROM table_name;
```

- **Entrainement :**

Utilisez la commande `SELECT*FROM` pour sélectionner la table “**employee_data**”.

2- Sélectionner une ou plusieurs colonnes :

Syntaxe :

```
SELECT column_1, column2
FROM table_name;
```

- Avec la commande SELECT, sélectionnez les **prénoms** et les **noms** des employées à partir de la table “**employees_data**”

WHERE:

Dans cette partie on va combiner la clause SELECT avec WHERE pour ajouter une condition pour extraire des données d’une partie de la BD.

Syntaxe :

```
SELECT column_1, column_2,... column_n
FROM table_name
WHERE condition;
```

Entrainement :

Sélectionnez toutes les personnes avec le prénom “**Charlie**”, “**Marilyn” , “Amy”** dans la table “**Employees_data**”

Equal Operators utilisés avec WHERE :

En SQL, il existe de nombreux autres mots-clés et symboles de liaison, appelés opérateurs, que vous pouvez utiliser avec la fonction WHERE :

Opérateur	L’inverse	Signification
AND	/	C’est comme dire "et aussi" : il faut que **toutes les conditions soient vraies**.
OR	/	Une seule condition suffit pour que ça fonctionne, c’est comme dire : Je veux voir les lignes qui respectent **au moins une des deux conditions**.”

IN	NOT IN	Onde mande si une valeur fait partie d'une liste. Par ex "Est-ce que c'est dans le lot ?"
LIKE	NOT LIKE	On cherche un **motif ou un bout de texte** dans une colonne. C'est pratique pour les prénoms par exemple.
BETWWEN... AND...	/	On vérifie si une valeur est **entre deux bornes**, un peu comme une date de début et de fin.
EXISTS	NOT EXISTS	
IS NULL	IS NOT NULL	On vérifie si **la case est vide (NULL)** ou si elle contient quelque chose.

😎 Focus sur `EXISTS`

L'opérateur `EXISTS` permet de dire :

> "Je veux voir les résultats seulement si une autre requête trouve quelque chose."

C'est un **test d'existence** sur une **sous-requête**.

Exemple

Vous avez deux tables :

- `employees` : la liste des employés
- `salaries` : les salaires des employés (avec une colonne `EmpID` qui correspond à l'identifiant employé)

Vous voulez afficher les employés qui ont bien un salaire enregistré :

```
SELECT *
FROM employees e
WHERE EXISTS (
    SELECT 1
    FROM salaries s
    WHERE s.EmpID = e.EmpID
);
```

En langage courant :

> "Je veux voir tous les employés... mais seulement ceux pour qui je trouve un salaire dans la table salaries."

- `EXISTS` = "Est-ce qu'il y a au moins **un résultat** dans la sous-requête ?"
- Si oui → la ligne principale est gardée.
- Si non → elle est ignorée.

AND :

Syntaxe :

```
SELECT column_1, column_2,... column_n
FROM table_name
WHERE condition_1 AND condition_2;
```

Entrainement :

sélectionnez les garçons qui s'appelle Charlie en utilisant AND statement

OR :

Syntaxe :

```
SELECT column_1, column_2,... column_n
FROM table_name
WHERE condition_1 OR condition_2;
```

Entrainement :

sélectionnez les garçons avec le prénom "**Jaiden**" ou le prénom "**Luke**" en utilisant la OR. Ressayez avec la clause WHERE...AND, que remarquez vous ?

OR vs AND:

AND : applicable quand on utilise des conditions sur différentes colonnes.

OR : applicable quand on utilise des conditions sur la même colonne.

Orde logique pour utiliser AND ou OR :

- SQL **traite d'abord les** `AND`, **puis** les `OR`.
- Donc, même si vous écriez `A AND B OR C`, SQL va d'abord l'interpréter comme :

  `(A AND B) OR C`

💡**Utilisez toujours des parenthèses** !

Syntaxe :

```
SELECT * FROM table_name
WHERE condition1 AND (condition2 OR condition3);
```

Entrainement :

Afficher les employés dont le **nom de famille est Smith**, et qui sont soit **hommes soit femmes**. (Oui, tous les genres, mais avec un filtre sur le nom.)

Exécutez les 2 codes ci-dessous :

Code 1 :

```
SELECT *
FROM employees
WHERE LastName = 'Smith' AND GenderCode = 'Male' OR GenderCode = 'Fe
```

Code 2 :

```
SELECT *
FROM employees
WHERE LastName = 'Smith' AND (GenderCode = 'Male' OR GenderCode = 'Fe
```

- **Que constatez vous ?**

IN:

La commande `IN` permet de filtrer les lignes **où une colonne correspond à une valeur parmi une liste.** C'est une manière plus simple et plus lisible que de faire plein de `OR`

Syntaxe :

```
SELECT * FROM table_name
WHERE column_name IN (value1, value2, ...);
```

Entrainement :

Utilisez la clause `IN` puis la clause `OR` pour afficher seulement la liste des personnes avec les prénoms : Harley, Frankie et Tylor.

- Comparez les 2 résultats (vous devez avoir le même résultat)
- Quelle clause permet d'avoir un code optimisé ?

NOT IN :

La commande `NOT IN` est l'inverse de `IN`. Elle permet de dire à SQL :

> "Je veux toutes les lignes sauf celles où la colonne contient ces valeurs-là."

Syntaxe :

```
SELECT * FROM table_name
WHERE column_name NOT IN (value1, value2, ...);
```

Entrainement :

Utilisez la clause NOT IN pour afficher tous les prénoms sauf Harley, Frankie et Taylor.

😇Astuce mémoire :

`IN` = est dans la liste

`NOT IN` = n'est pas dans la liste

LIKE - NOT LIKE :

`LIKE` permet de rechercher un **motif de texte** (pattern) dans une colonne, souvent avec

- `%` = n'importe quel nombre de caractères
- `_` = un seul caractère

Entrainement :

Exécutez les codes ci-dessous et notez vos observations :

```
-- Tous les prénoms qui commencent par "Char"
SELECT * FROM employees
WHERE FirstName LIKE 'Char%';
```

```
-- Tous les prénoms qui se terminent par "ie"
SELECT * FROM employees
WHERE FirstName LIKE '%ie';
```

```
-- Tous les prénoms contenant "son"
SELECT * FROM employees
WHERE FirstName LIKE '%son%';
```

```
-- Tous les prénoms de 5 lettres qui commencent par "El"
SELECT * FROM employees
WHERE FirstName LIKE 'El___';
```

BETWEEN...AND :

Syntaxe :

```
SELECT * FROM table_name
WHERE column_name BETWEEN 'valeur1' AND 'valeur2';
```

Entrainement :

Afficher les personnes embauchées **entre le 1er janvier 2020 et le 31 décembre 2021**.

IS NOT NULL :

Syntaxe :

```
SELECT column_1, column_2, ..., column_n
FROM table_name
WHERE column_name IS NOT NULL;
```

- Appliquez la clause IS NOT NULL sur la table employees pour voir s'il y a des prénoms qui manque pas.

FONCTIONS D'AGGRÉGATION :

Les **fonctions d'agrégation** te permettent de **faire des calculs globaux** sur des colonnes :

- compter
- additionner
- chercher le min, le max
- faire une moyenne

On les utilise souvent avec `SELECT` pour **résumer les données**.

Fonction	Ce qu'elle fait	Exemple simple
`COUNT()`	Compte le nombre de lignes	Combien d'employés au total
`SUM()`	Fait la somme	Additionner tous les salaires (si dispo)
`MIN()`	Cherche la valeur la plus basse	La plus vieille date d'embauche
`MAX()`	Cherche la valeur la plus haute	La date d'embauche la plus récente
`AVG()`	Calcule la moyenne	Moyenne des notes, des salaires, etc.

Entrainement:

- Combien d'employées sont enregistrés dans la BD ?
- Combien de noms différents peut-on trouver dans la table "employés" ? (Utilisez la commande Count DISTINCT).

ORDER BY:

Entrainement

- Triez la table "employees" par prénoms dans un ordre ascendant.
- Triez la table "employees" par prénoms dans un ordre descendant.

GROUP BY

La commande GROUP BY est utilisée en SQL pour grouper plusieurs résultats et utiliser une fonction de totaux sur un groupe de résultat. Sur une table qui contient toutes les ventes d'un magasin, il est par exemple possible de liste regrouper les ventes par clients identiques et d'obtenir le coût total des achats pour chaque client.

Syntaxe:

```
SELECT column1, column2, ..., aggregate_function(column_name)
FROM table_name
WHERE condition
GROUP BY column1, column2, ...;
```

Entrainement :

Testez les codes ci-dessous :

Code 1:

```
SELECT first_name
FROM employees
```

```
GROUP BY first_name;
```

Code 2: affiche le nombre d'apparition de chaque prénom

```
SELECT
COUNT(first_name)
FROM
employees
GROUP BY first_name;
```

Code 3 :

```
SELECT first_name, COUNT(first_name)
FROM
employees
GROUP BY first_name;
```

- Quelle est la différence entre les 3 codes ?

Code 4:

```
SELECT first_name, COUNT(first_name)
FROM employees
GROUP BY first_name
ORDER BY first_name DESC;
```

- Expliquez le code 4.

HAVING :

Syntaxe:

```
SELECT column1, column2, ..., aggregate_function(column_name)
FROM table_name
WHERE condition
GROUP BY column1, column2, ...
HAVING condition;
```

Entrainement :

- Exécutez le code ci-dessous et analysez les résultat (à adaptez en fonction de votre csv)

```
SELECT
    first_name, COUNT(first_name) AS names_count
FROM
    employees
GROUP BY
    first_name
HAVING
    COUNT(first_name) > 250
ORDER BY
    first_name;
```

- Quel est le rôle de la clause HAVING ?

WHERE vs HAVING :

- Quelle est la différence entre HAVING et WHERE ?

AS (alias)

Dans le langage SQL il est possible d'utiliser des **alias** pour renommer temporairement une colonne ou une table dans une requête. Cette astuce est particulièrement utile pour faciliter la lecture des requêtes.

Syntaxe :

Alias d'une table :

```
SELECT column1, column2
FROM table_name AS alias_name;
```

Alias d'une colonne:

```
SELECT column_name AS alias_name
```

```
FROM table_name;
```

Alias d’une table et colonne dans la même requête :

```
SELECT t1.column1 AS alias1, t2.column2 AS alias2
FROM table1 AS t1
JOIN table2 AS t2 ON t1.id = t2.id;
```

Entrainement :

Exécutez le code ci-dessous, vous devriez avoir en en-tête : first_name et Count(first_name) :

```
SELECT
   first_name,
   COUNT(first_name)
FROM
   employees
GROUP BY
   first_name
ORDER BY
   first_name;
```

Pour faciliter l’accès à l’information et améliorer le visuel des BDD on va utiliser les Alias, exécutez maintenant le code ci-dessous :

```
SELECT
   first_name,
   COUNT(first_name) AS names_count
FROM
   employees
GROUP BY
   first_name
ORDER BY
   first_name;
```

Vous obtenez de nouveaux en-tête.

Bonus - Entrainement en ligne :

https://www.w3schools.com/sql/sql_exercises.asp

Livrables :

Aucun

Évaluation

Évaluation orale individuelle