**Les Bases de Données**

**Le Language SQL - Consultation**

**Formatrice : Florence CALMETTES**

**Objectif :**

L’objectif de ce support est de vous donner l’ensemble des commandes SQL que vous aurez besoin dans les exercices suivants.

Vous pouvez vous appuyer également sur le site Internet : <https://sql.sh/>

Il existe 3 types de requêtes :

* Les requêtes de structure
* Les requêtes de mise à jour de la base de données
* Les requêtes de consultation

**Prérequis : Installation de MySQL WorkBench**

**Attention :**

Quand de grosses opérations doivent être réalisées sur une base de données qui est en production (mise à jour, modification, suppression …) il est impératif de :

* Réaliser les opérations sur une base de test avant
* Sauvegarder la base de données avant de faire les opérations

**Préparation de l’environnement de travail :**

Démarrer MySQL WorkBench

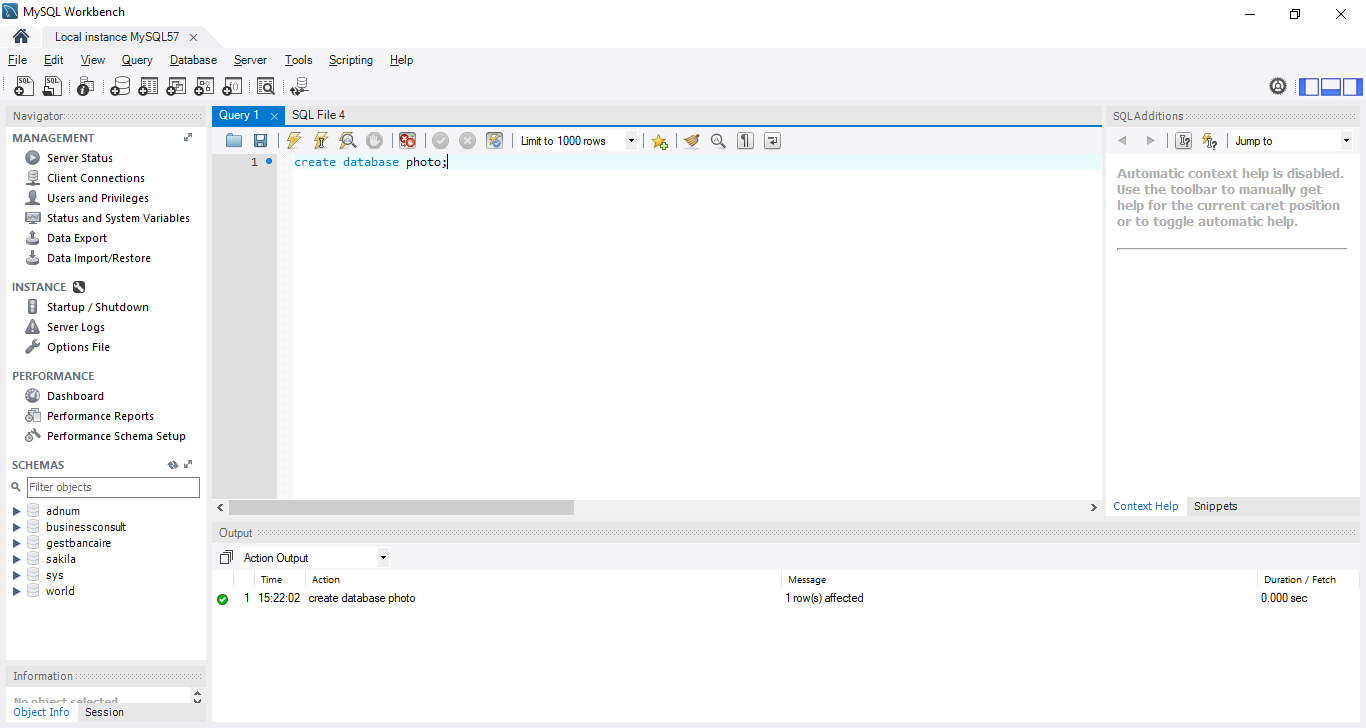
**Création de la Base de Données :**

**CREATE DATABASE** nom\_base\_de\_donnée;

Reprenons notre exemple :

**CREATE DATABASE** photo;

Pour exécuter une requête, faire l’éclair jaune



**Requêtes de Consultation :**

**Select :**

La commande SELECT permet de faire des opérations de recherche et de calcul à partir des enregistrements d’un ou plusieurs attributs d’une table.

**Syntaxe de la commande SELECT :**

SELECT nom\_attribut

FROM table1 ;

Cette requête va sélectionner (SELECT) l’attribut « nom\_attribut » provenant de la table (FROM) table1

Exemple :

use photo;

select nom\_personne

from PERSONNE ;

**Pour sélectionner plusieurs attributs on les sépare par des virgules :**

SELECT nom\_attribut1, nom\_attribut2

FROM table1 ;

Exemple :

use photo;

select nom\_lieux, ville\_lieux

from lieux ;

**Pour sélectionner tous les attributs d’une table :**

SELECT \*

FROM table1 ;

Exemple :

use photo;

select \*

from PERSONNE ;

**Distinct :**

La commande SELECT peut potentiellement afficher des enregistrements en double.

La commande DISTINCT permet d’éviter des redondances dans les résultats.

**Syntaxe de la commande DISTINCT :**

SELECT DISTINCT nom\_attribut

FROM table1 ;

Cette requête va sélectionner (SELECT) l’attribut nom\_attribut provenant de la table (FROM) table1, en évitant de retourner des doublons.

Exemple :

Création de 3 photos ayant le même nom :

use photo;

insert into photo (nom\_photo)

values ("photo1"),

("photo1"),

("photo1"),

("photo2");

Ecrire une requête qui affiche les noms des photos sans doublon.

use photo;

select distinct nom\_photo

from photo;

Résultat :

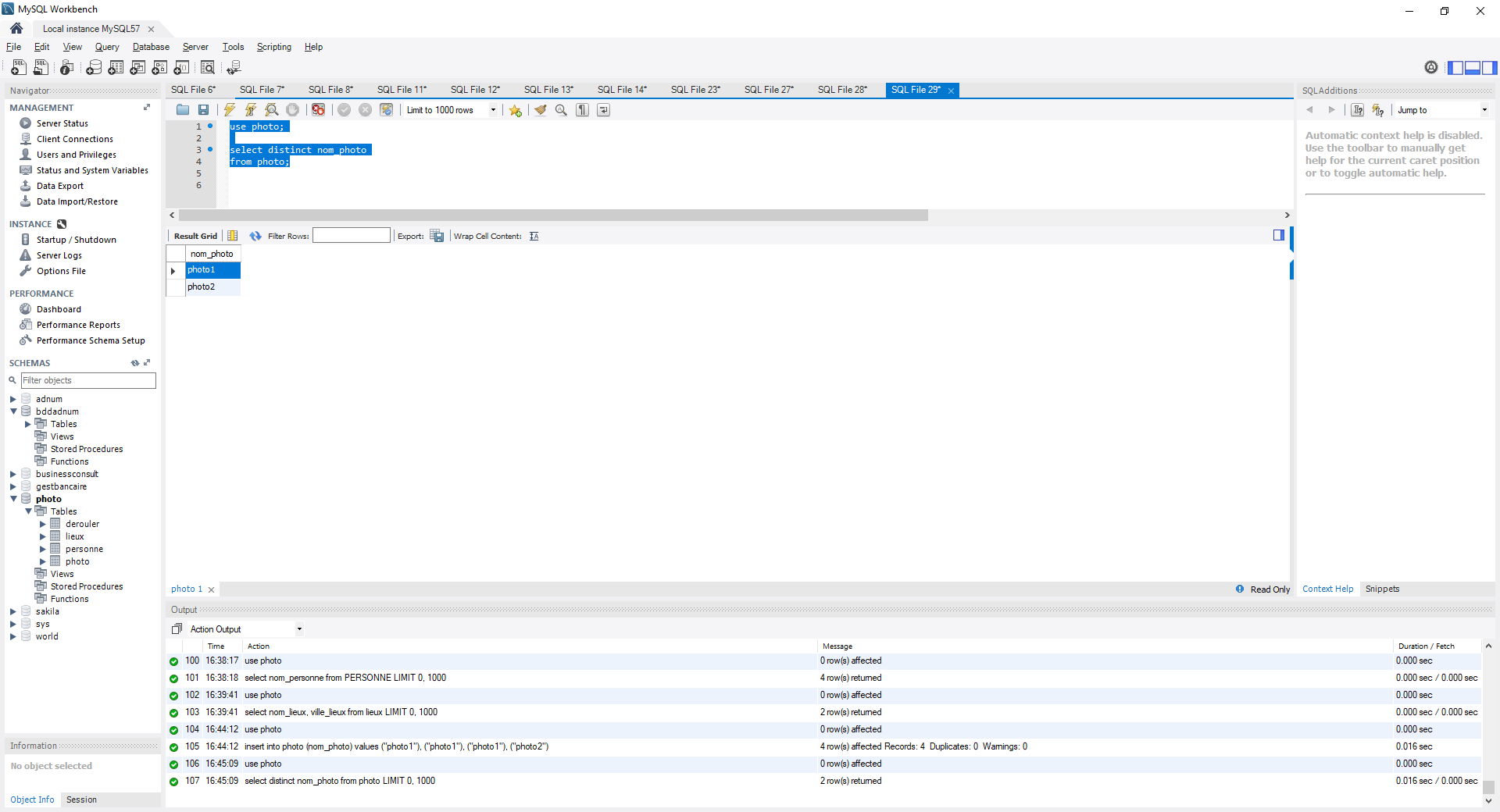


Photo 1 est créé 3 fois mais apparait qu’une fois grâce au distinct.

**As :**

La commande AS permet d’utiliser des alias pour renommer temporairement un attribut ou une table dans une requête.

Cela permet de facilité la lecture des requêtes.

A l’exécution la colonne prendra le nom de l’alias.

**Syntaxe pour renommer un attribut temporairement:**

SELECT nom\_attribut1 AS na1, nom\_attribut2 AS na2

FROM table1 ;

**Il est possible aussi de l’écrire ainsi :**

SELECT nom\_attribut1 na1, nom\_attribut2 na2

FROM table1 ;

Il est préférable d’utiliser la commande AS pour faciliter la lecture.

**Commande pour renommer une table :**

SELECT nom\_attribut1

FROM table1 AS t1 ;

**Exemple :**

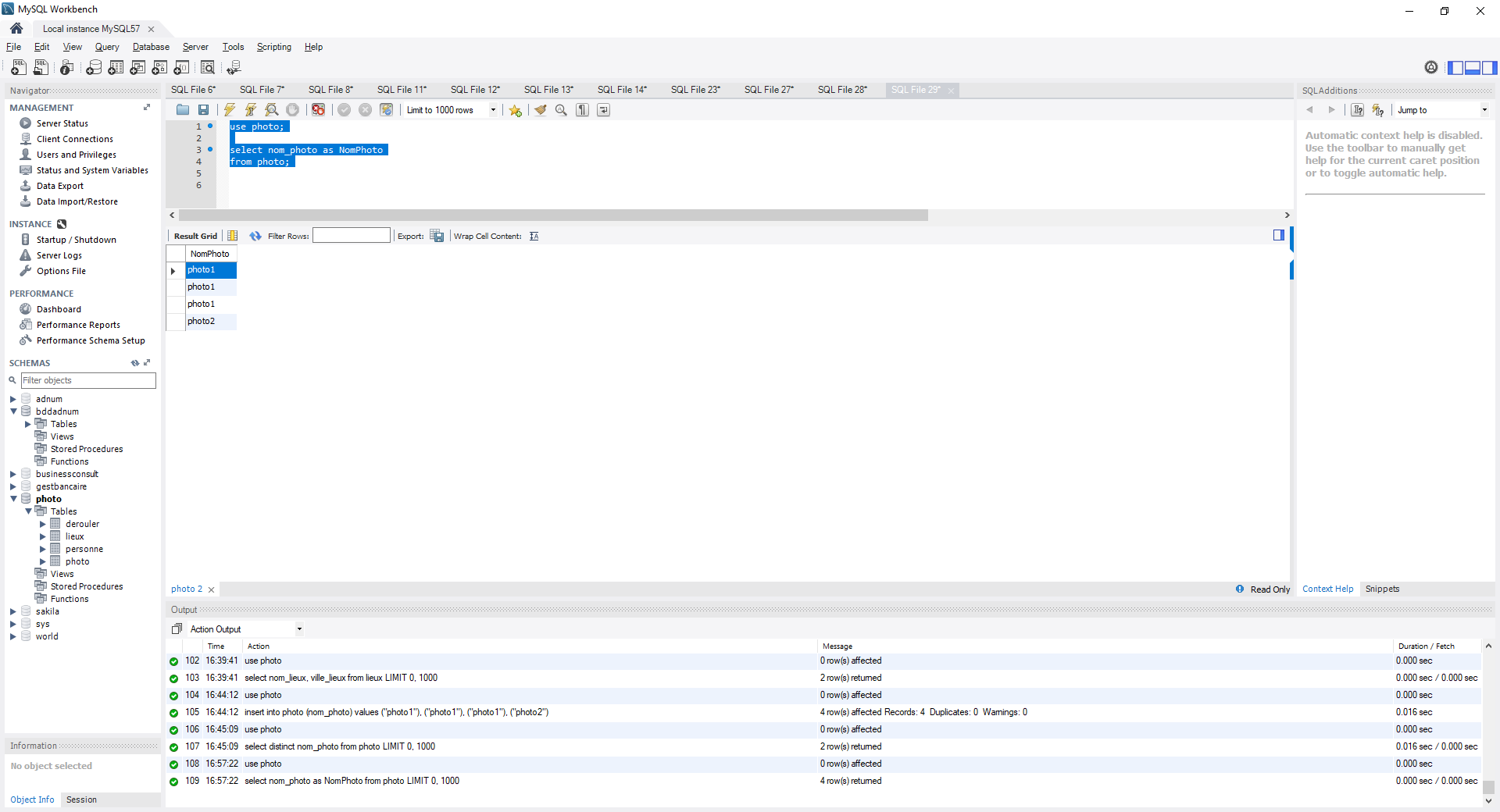
use photo;

select nom\_photo as NomPhoto

from photo;

Résultat :

La colonne porte le nom de l’Alias :



**Where :**

La commande WHERE permet d’extraire des enregistrements d’une base de données qui respectent une condition.

Cela permet d’obtenir uniquement les informations désirées.

**Syntaxe de la commande WHERE :**

SELECT nom\_attribut1, nom\_attribut2

FROM table1

WHERE condition;

Les opérateurs de comparaisons :

|  |  |
| --- | --- |
| **Opérateur** | **Description** |
| = | Egale |
| <> | Pas égale |
| != | Pas égale |
| > | Supérieur à |
| < | Inférieur à |
| >= | Supérieur ou égal à |
| <= | Inférieur ou égal à |
| IN | Liste de plusieurs valeurs possibles |
| BETWEEN …. AND | Valeur comprise dans un intervalle de données |
| LIKE | Recherche en spécifiant le début, le milieu ou la fin d’un mot |
| IS NULL | Valeur est nulle |
| IS NOT NULL | Valeur n’est pas nulle |

**Exemple :**

**Donner la liste des photos dont le nom est « photo1 »**

use photo;

select \*

from photo

Where nom\_photo="photo1";

**Donner la liste des photos dont le nom est entre A et C**

use photo;

select \*

from photo

Where nom\_photo between "A" and "C";

**And / Or :**

Les opérateurs logiques AND et OR peuvent être utilisés dans la commande WHERE pour combiner des conditions.

**Syntaxe de la commande AND :**

SELECT nom\_attribut1, nom\_attribut2

FROM table1

WHERE condition1 AND condition2;

L’opérateur And permet d’afficher un résultat que si les conditions 1 et 2 sont vraies.

**Syntaxe de la commande OR :**

SELECT nom\_attribut1, nom\_attribut2

FROM table1

WHERE condition1 OR condition2;

L’opérateur Or permet d’afficher un résultat si la condition 1 ou 2 est vraie.

**Il est possible de combiner les deux :**

SELECT nom\_attribut1, nom\_attribut2

FROM table1

WHERE condition1 AND (condition2 OR condition3);

**Order By :**

La commande ORDER BY permet de trier des lignes dans un résultat d’une requête.

Il est possible de trier les données sur un ou plusieurs attributs par ordre ascendant ou descendant.

**Syntaxe de la commande ORDER BY** :

SELECT nom\_attribut1, nom\_attribut2

FROM table1

ORDER BY nom\_attribut1;

Par défaut, les résultats sont classés par ordre ascendant.

Pour trier le résultat par ordre décroissant nous utilisons le suffixe DESC :

SELECT nom\_attribut1, nom\_attribut2

FROM table1

ORDER BY nom\_attribut1 DESC, nom\_attribut2 ASC ;

**Group by :**

La commande GROUP BY est utilisée avec les fonctions de calcul.

Elle permet de regrouper le résultat.

Elle est obligatoire sur tous les attributs qui entourent la fonction de calcul dans le SELECT.

**Syntaxe de la commande GROUP BY :**

SELECT nom\_attribut1, fonction (nom\_attribut2)

FROM table1

GROUP BY nom\_attribut1;

Cette commande doit toujours s’utiliser après la commande WHERE et avant la commande HAVING.

Les fonctions possibles :

|  |  |
| --- | --- |
| **Fonctions** | **Description** |
| AVG() | Calcul de la moyenne d’un ensemble de valeurs |
| COUNT() | Pour compter le nombre de lignes concernées |
| MAX() | Permet de récupérer la plus haute valeur |
| MIN() | Permet de récupérer la plus petite valeur |
| SUM() | Somme de plusieurs lignes |

**Having :**

La commande HAVING est presque similaire au WHERE à la seule différence que le HAVING permet de filtrer en utilisant les fonctions de calcul.

**Syntaxe de la commande HAVING :**

SELECT nom\_attribut1, fonction (nom\_attribut2)

FROM table1

GROUP BY nom\_attribut1

HAVING fonction (nom\_attribut2) = condition1;

**Récapitulatif du Select :**

Récapitulatif de l’ordre d’utilisation des commandes :

**SELECT** [DISTINCT] liste des champs, calcul

**FROM** table

[ **WHERE** formule de restriction ]

[ **GROUP BY** liste des champs ]

[ **HAVING** formule de restriction ]

[ **ORDER BY** liste des champs ]

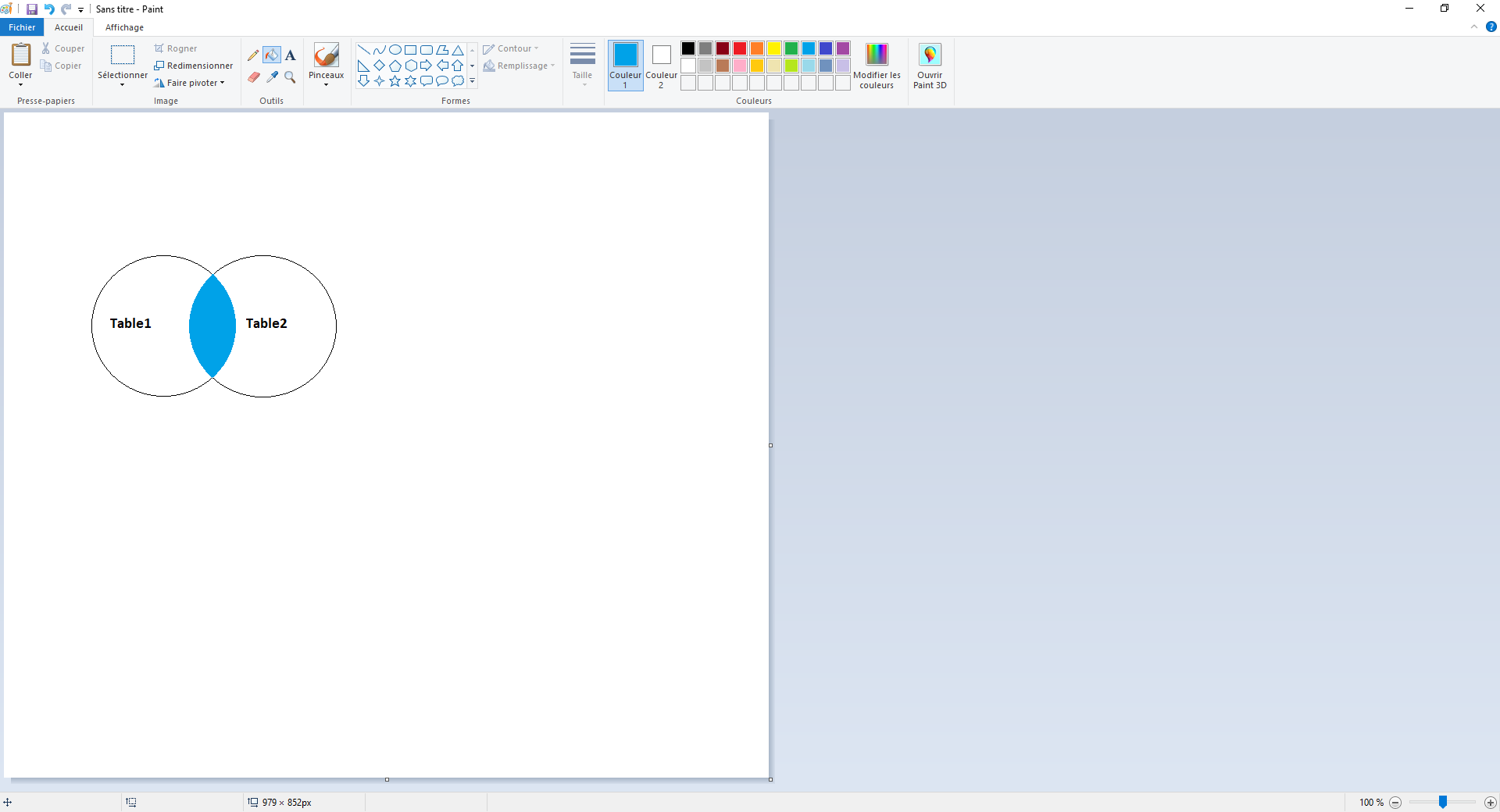
**Jointures :**

Les jointures permettent d’associer plusieurs tables dans une requête par la clé étrangère.

Types de jointure :

* INNER JOIN
* LEFT JOIN
* RIGHT JOIN
* FULL JOIN

**Inner Join :**



Cette commande retourne les enregistrements quand la condition est vraie dans les deux tables.

Elle retournera tous les enregistrements qui se trouvent dans la table 1 et qui ont forcément un lien (clé étrangère not null) avec la table 2.

Syntaxe de la commande INNER JOIN :

SELECT \*

FROM table1

INNER JOIN table2

ON table1.id = table2.id;

Cette requête stipule qu’il faut sélectionner tous les enregistrements des tables table1 et table2 lorsque l’attribut id de la table1 est égal aux données de l’attribut id de la table2.

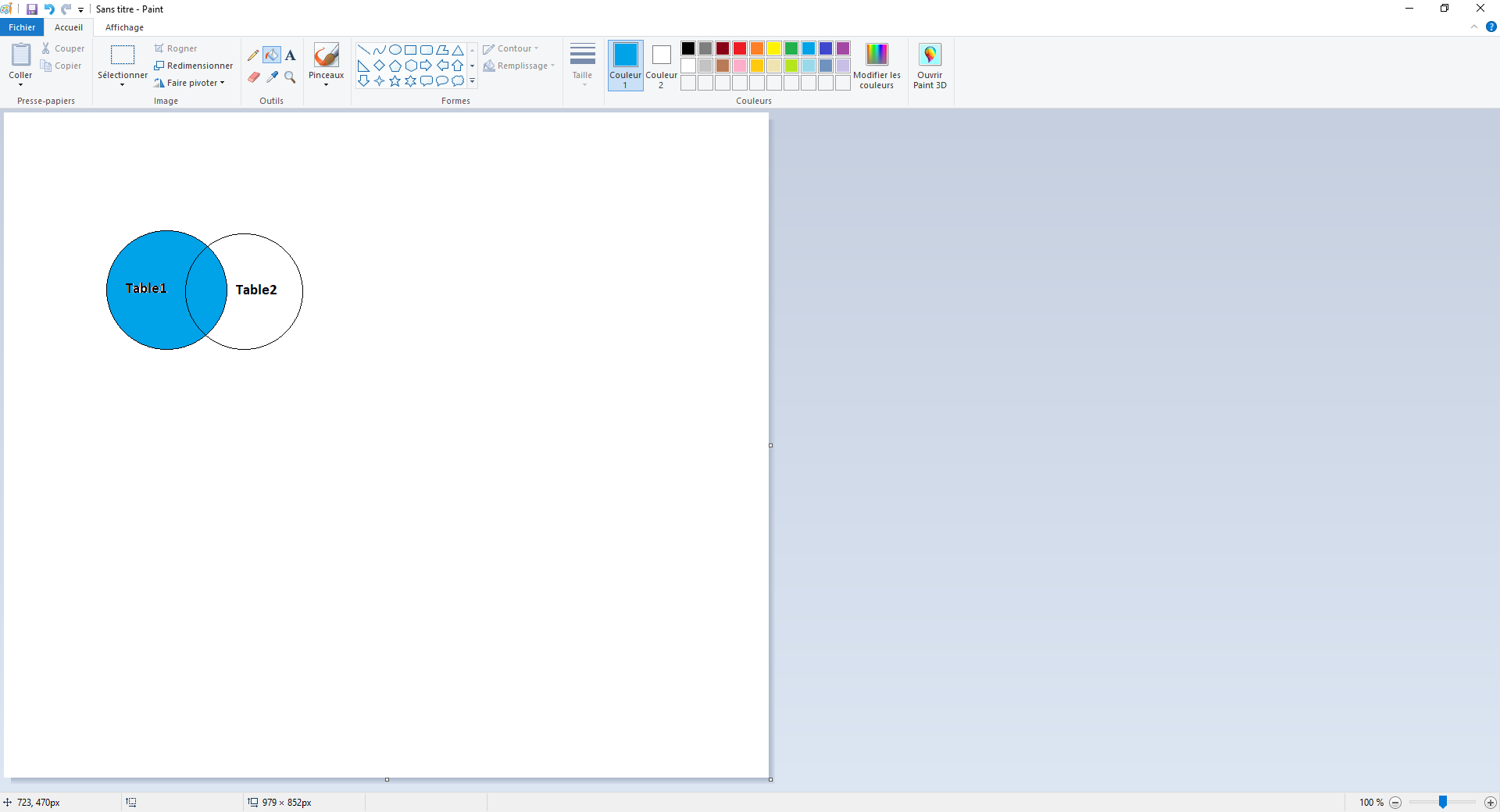
Nous pouvons aussi l’écrire :

SELECT \*

FROM table1, table2

WHERE table1.id = table2.id ;

**Left Join :**



C’est une jointure externe gauche.

Cette commande retourne tous les enregistrements de la table 1 même si la condition n’est pas vérifiée dans l’autre table.

Elle retournera tous les enregistrements de la table 1 qu’ils aient un lien ou non avec la table 2.

Syntaxe de la commande LEFT JOIN :

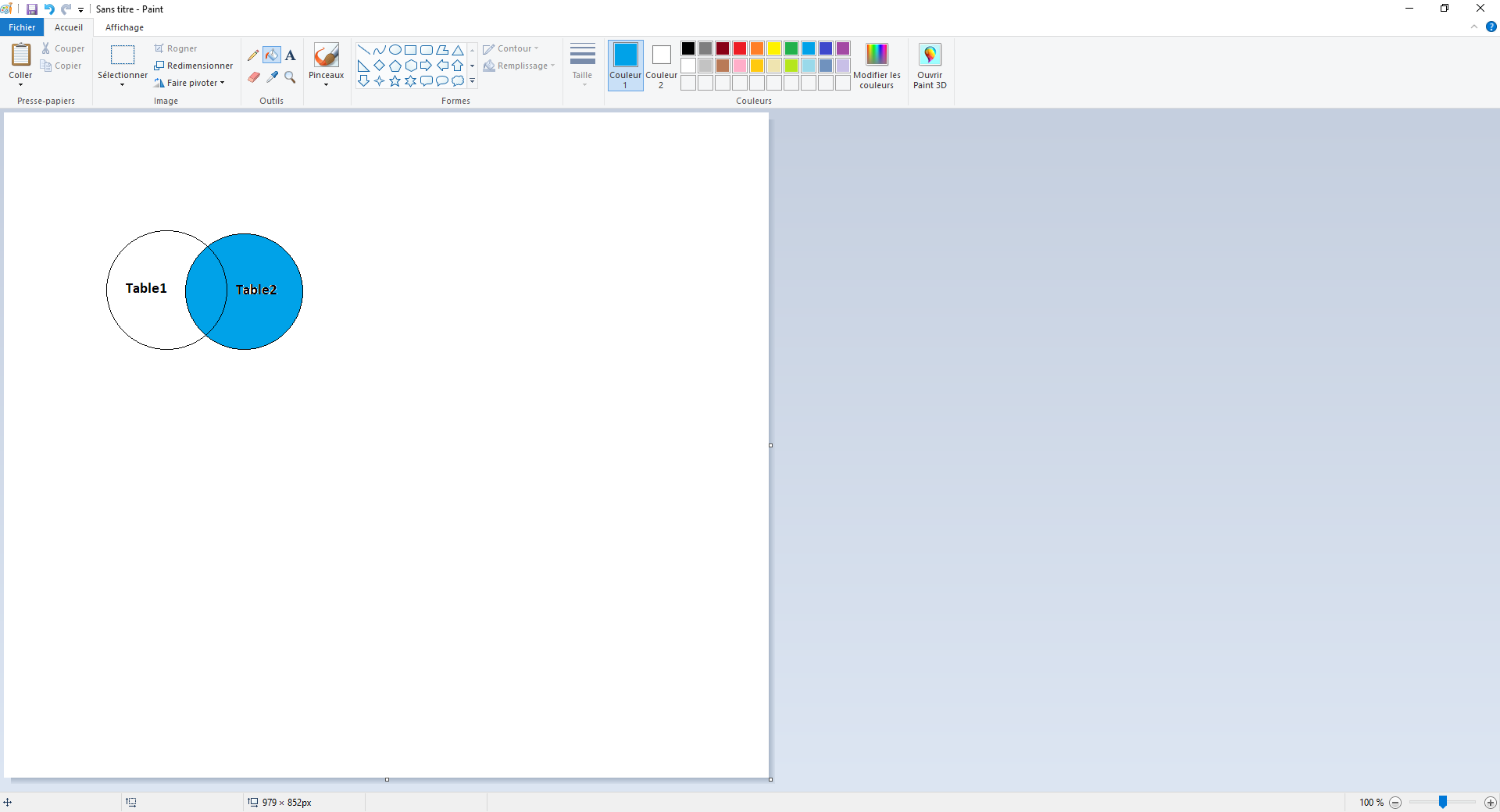
SELECT \*

FROM table1

LEFT JOIN table2

ON table1.id = table2. id ;

**Right Join :**



C’est une jointure externe droite.

Cette commande retourne tous les enregistrements de la table 2 même si la condition n’est pas vérifiée dans l’autre table.

Elle retournera tous les enregistrements de la table 2 qu’ils aient un lien ou non avec la table 1.

Syntaxe de la commande RIGHT JOIN :

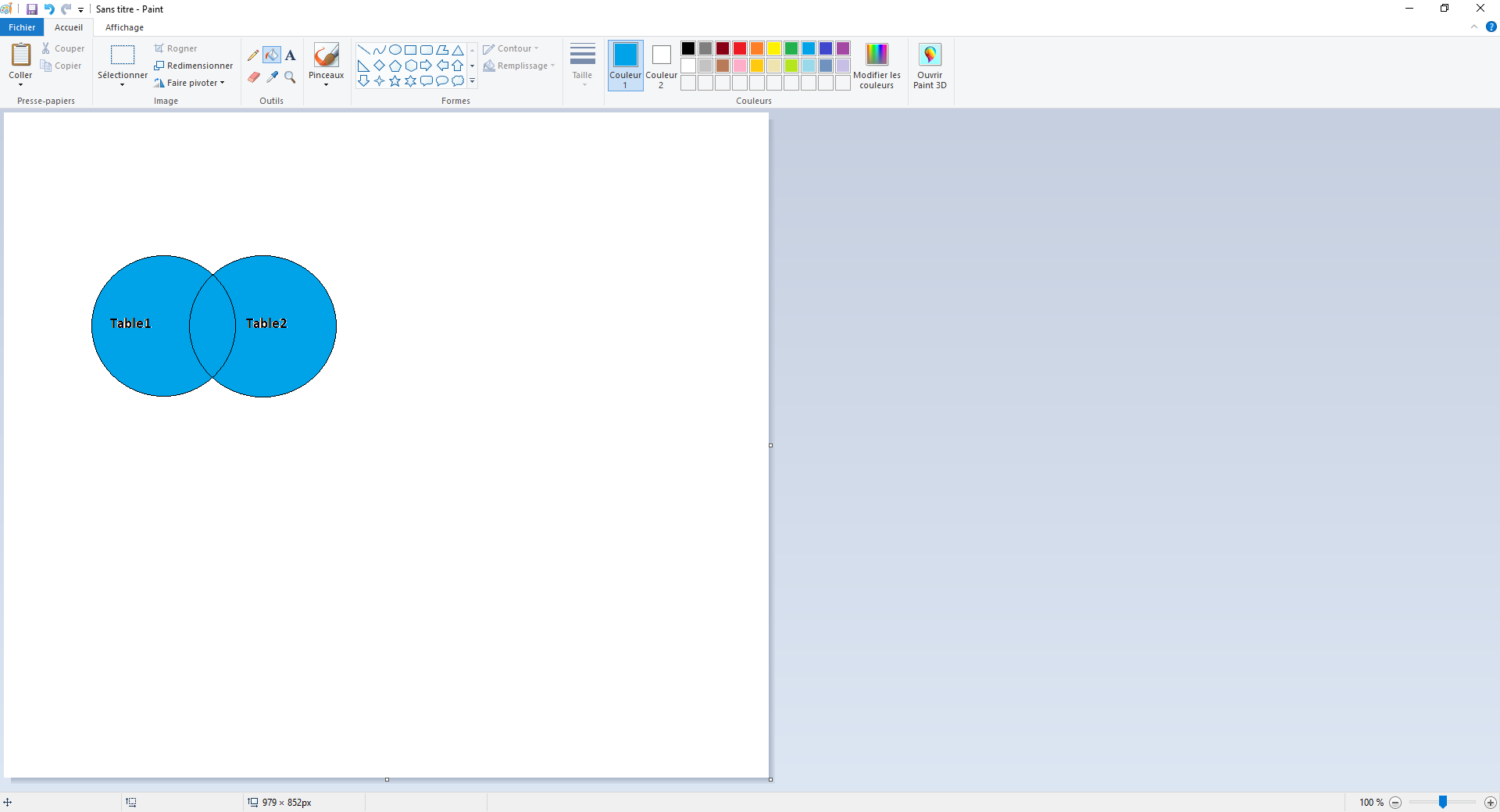
SELECT \*

FROM table1

RIGHT JOIN table2

ON table1.id = table2.id ;

**Full Join :**



Permet de combiner les résultats de deux tables.

Syntaxe de la commande FULL JOIN :

SELECT \*

FROM table1

FULL JOIN table2

ON table1.id = table2.id ;

Affichera tous les enregistrements de la table 1 et 2 et associera les lignes qui ont table1.id et table2.id en commun.

**Sous requêtes :**

Une sous requête consiste à exécuter une requête à l’intérieur d’une autre requête.

Une requête imbriquée est souvent utilisée au sein d’une clause WHERE ou HAVING pour remplacer une ou plusieurs constantes.

**Syntaxe d’une sous requête :**

SELECT \*

FROM table1

WHERE nom\_attribut1 IN (

SELECT nom\_attribut2

FROM table2

WHERE nom\_attribut2= " valeur"

) ;

La sous requête renvoie la valeur d’un champ.

La requête principale va chercher les résultats de la tables1 dont l’attribut de la table 1 est filtré sur un ensemble de valeurs renvoyées par la sous requête.