



# Manipuler les Bases de données avec le SQL



# Le parcours

Découvrir différents métiers et leurs composantes au travers des présentations et de la pratique personnelle

# Manipuler les Bases de données avec le SQL

## Sommaire :

Les bases de données

Le SGBD

Les bases de données les plus courantes

Structure d'une table de base de données

Les SGBDR les plus populaires

Installer un SGBD local

Lire les donnée d'une base de donnée

Créer des tables et Insérer des enregistrements

Modifier les données et les colonnes

Supprimer des données, des tables et les colonne

# Les bases de données

- Une base de données est une collection d'informations organisées et structurées afin d'être facilement consultables, gérables et mises à jour.
- Une base de données permet à l'utilisateur de réaliser les opérations : récupérer, ajouter, modifier ou supprimer des données.
- CREATE READ UPDATE DELETE -> CRUD .

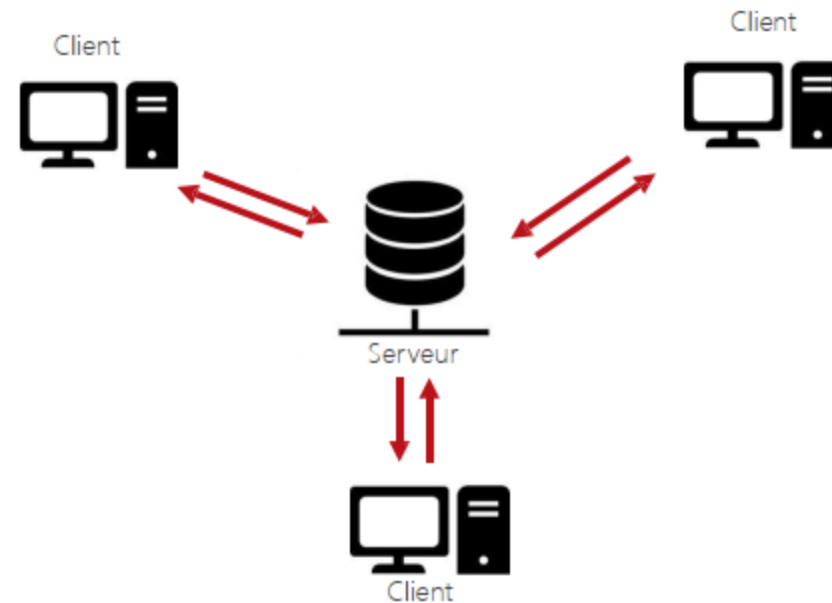


# Les bases de données

## Système de Gestion de Bases de Données (SGBD)

**Le SGBD est le logiciel qui va vous permettre de manipuler les données d'une base.**

C'est un outil permettant plusieurs utilisateurs simultanés : partage des données



# Les bases de données

## Les 2 modèles bases de données les plus courantes :

les bases de données relationnelles,  
i.e. données sous forme de **tables**,  
Communication à la base de donnée  
avec le SQL :  
**S**tructured **Q**uery **L**anguage.

les bases de données noSQL, les données  
ne sont pas sous forme de tables mais sous  
forme de **clé-valeur**, (ex : HBASE, MongoDB)  
Enregistrement en JSON  
**J**ava**S**cript **O**bject **N**otation

Chaque base NoSQL peut avoir sa propre  
syntaxe de Communication de base de  
données.

# Les bases de données

## Structure d'une table dans un Base de données Relationnelle

**Table also called Relation**

Primary Key

Domain  
Ex: NOT NULL

© guru99.com

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

**Tuple OR Row**

Total # of rows is **Cardinality**

**Column OR Attributes**

Total # of column is **Degree**



# Les bases de données

## Les SGBDR les plus populaires

- **MySQL**

C'est le plus connu et utilisé, car il était (auparavant) open-source avant d'être racheté par Oracle.



- **Oracle Database**

C'est très cher, mais utile pour traiter un très gros volume de données. C'est une solution pour les très grandes entreprises.



- **PostgreSQL**

C'est "l'autre" grand SGBD open-source disponible sur le marché.



- **MariaDB**

C'est une « copie » améliorée de MySQL créée par le fondateur de MySQL lui-même. C'est un SGBD open-source qui devient de plus en plus populaire.





# Le SQL : Structured Query Language

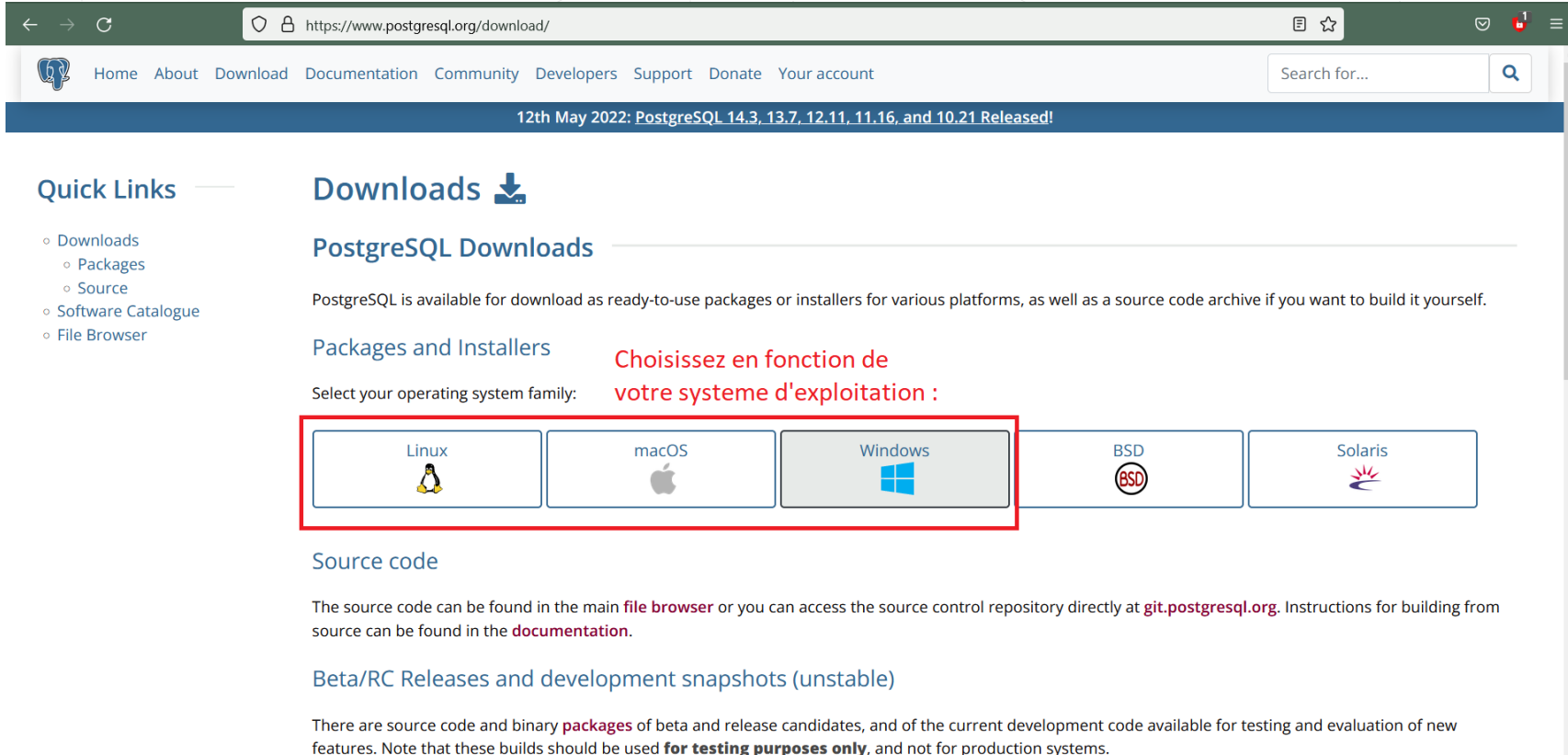
**Le langage SQL est un langage simple qui permet de travailler sur une base de données**

Il va nous permettre d'ajouter, lire, modifier ou supprimer des informations d'une base de donnée.

Pour pouvoir s'exercer, nous aurons besoin :

- un SGBD : nous pouvons utiliser : PostgreSQL  
<https://www.postgresql.org/download/>
- un outil de manipulation de la base de donnée :  
il existe une extension dans VSCode


# Installer PostgreSQL



The screenshot shows the PostgreSQL download page. The browser address bar displays `https://www.postgresql.org/download/`. The navigation bar includes links for Home, About, Download, Documentation, Community, Developers, Support, Donate, and Your account. A search bar is located on the right. A blue banner at the top of the page content area reads: "12th May 2022: PostgreSQL 14.3, 13.7, 12.11, 11.16, and 10.21 Released!".

**Quick Links**

- Downloads
  - Packages
  - Source
- Software Catalogue
- File Browser

**Downloads** 






## PostgreSQL Downloads

PostgreSQL is available for download as ready-to-use packages or installers for various platforms, as well as a source code archive if you want to build it yourself.

**Packages and Installers**

Choisissez en fonction de votre système d'exploitation :

Select your operating system family:

Linux 	macOS 	<b>Windows </b>	BSD 	Solaris 
--	--	--	--	--

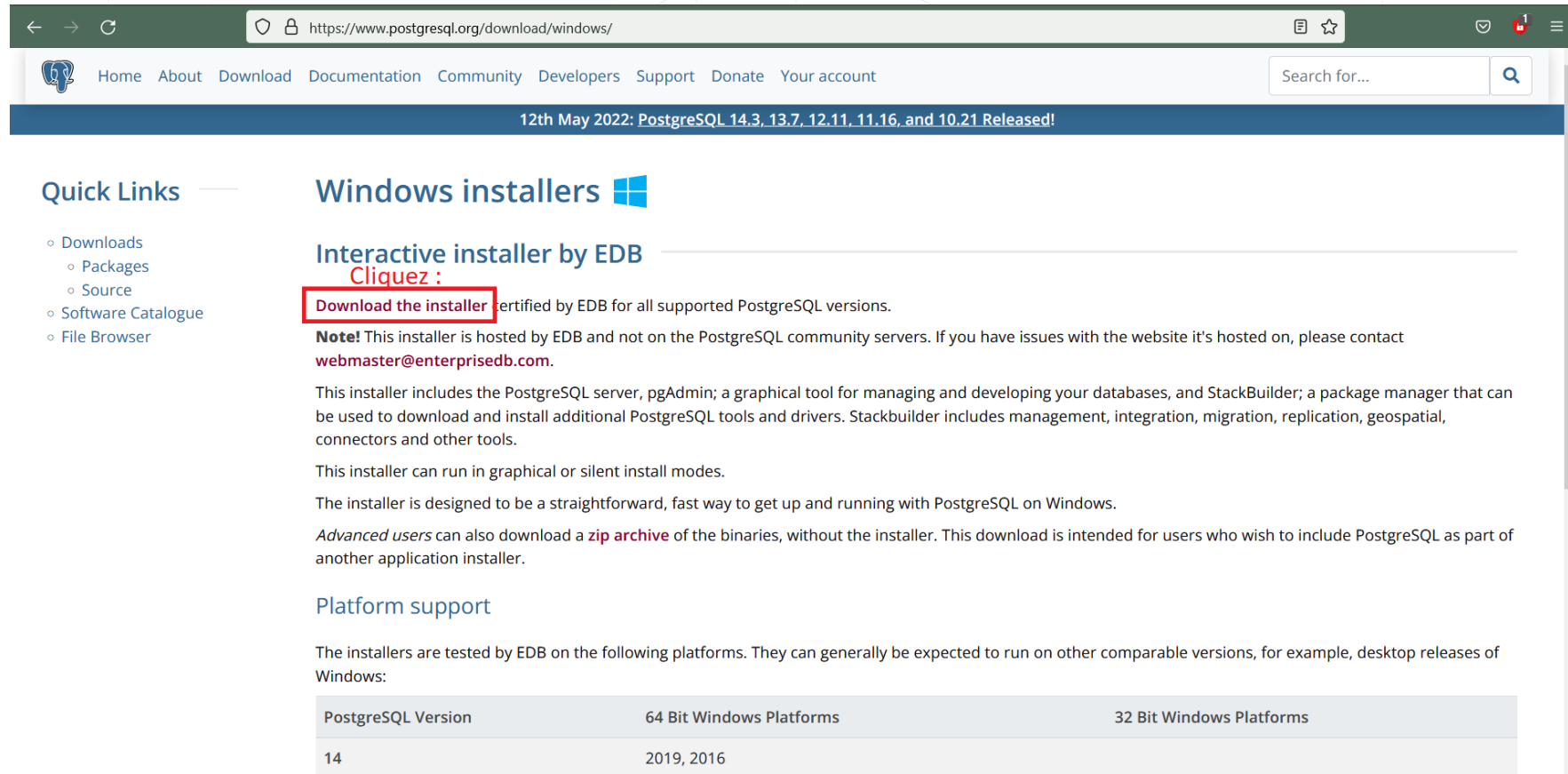
**Source code**

The source code can be found in the main **file browser** or you can access the source control repository directly at [git.postgresql.org](https://git.postgresql.org). Instructions for building from source can be found in the **documentation**.

**Beta/RC Releases and development snapshots (unstable)**

There are source code and binary **packages** of beta and release candidates, and of the current development code available for testing and evaluation of new features. Note that these builds should be used **for testing purposes only**, and not for production systems.


# Installer PostgreSQL



The screenshot shows the PostgreSQL website's download page for Windows. The browser address bar shows the URL <https://www.postgresql.org/download/windows/>. The page has a navigation bar with links: Home, About, Download, Documentation, Community, Developers, Support, Donate, and Your account. A search bar is also present. A blue banner at the top of the page content area reads: "12th May 2022: PostgreSQL 14.3, 13.7, 12.11, 11.16, and 10.21 Released!".

**Quick Links**

- Downloads
  - Packages
  - Source
- Software Catalogue
- File Browser

**Windows installers** 

**Interactive installer by EDB**

**Cliquez :**

**Download the installer** certified by EDB for all supported PostgreSQL versions.

**Note!** This installer is hosted by EDB and not on the PostgreSQL community servers. If you have issues with the website it's hosted on, please contact [webmaster@enterprisedb.com](mailto:webmaster@enterprisedb.com).

This installer includes the PostgreSQL server, pgAdmin; a graphical tool for managing and developing your databases, and StackBuilder; a package manager that can be used to download and install additional PostgreSQL tools and drivers. Stackbuilder includes management, integration, migration, replication, geospatial, connectors and other tools.

This installer can run in graphical or silent install modes.

The installer is designed to be a straightforward, fast way to get up and running with PostgreSQL on Windows.

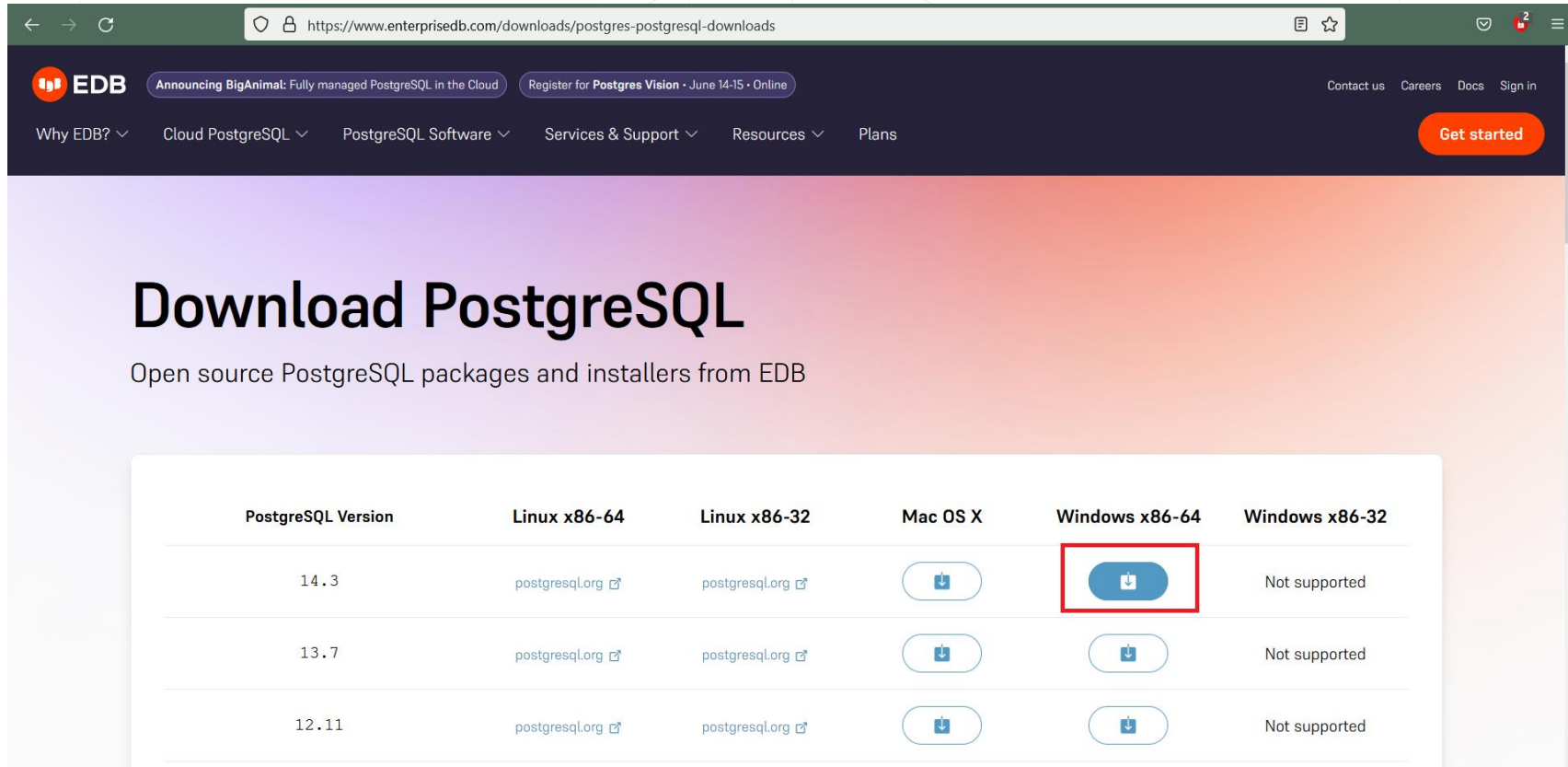
*Advanced users* can also download a **zip archive** of the binaries, without the installer. This download is intended for users who wish to include PostgreSQL as part of another application installer.

**Platform support**

The installers are tested by EDB on the following platforms. They can generally be expected to run on other comparable versions, for example, desktop releases of Windows:

PostgreSQL Version	64 Bit Windows Platforms	32 Bit Windows Platforms
14	2019, 2016	

# Installer PostgreSQL



The screenshot shows the EDB PostgreSQL download page. The main heading is "Download PostgreSQL" with the subtitle "Open source PostgreSQL packages and installers from EDB". Below this is a table listing PostgreSQL versions and their corresponding download links for different operating systems. The "Windows x86-64" column for version 14.3 is highlighted with a red box.

PostgreSQL Version	Linux x86-64	Linux x86-32	Mac OS X	Windows x86-64	Windows x86-32
14.3	<a href="#">postgresql.org</a>	<a href="#">postgresql.org</a>	<a href="#">Download</a>	<a href="#">Download</a>	Not supported
13.7	<a href="#">postgresql.org</a>	<a href="#">postgresql.org</a>	<a href="#">Download</a>	<a href="#">Download</a>	Not supported
12.11	<a href="#">postgresql.org</a>	<a href="#">postgresql.org</a>	<a href="#">Download</a>	<a href="#">Download</a>	Not supported

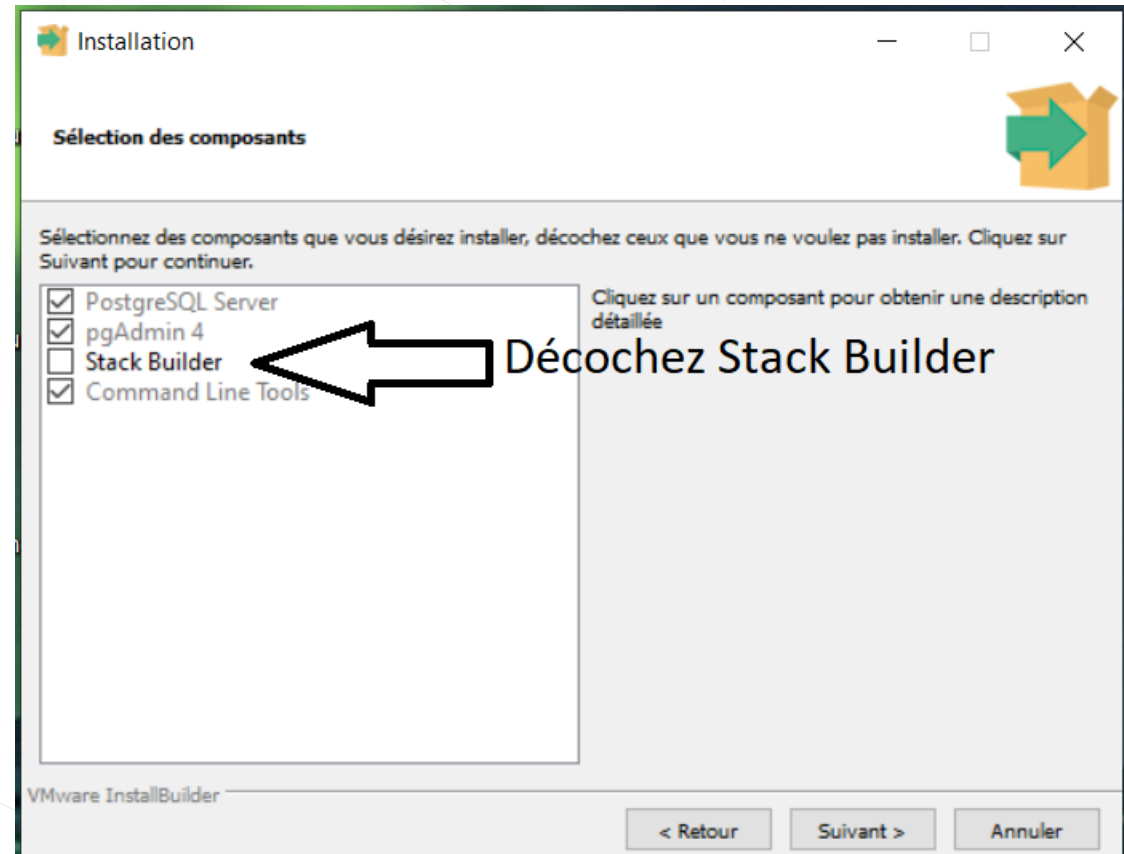
# Installer PostgreSQL

A l'installation, faites bien en sorte de décocher Stack Builder dans la liste :  
Sélection des composants

Lors du mot passe, choisissez un mot de passe facile à retenir (admin)

Laissez le port par défaut

Pour le reste cliquez sur suivant

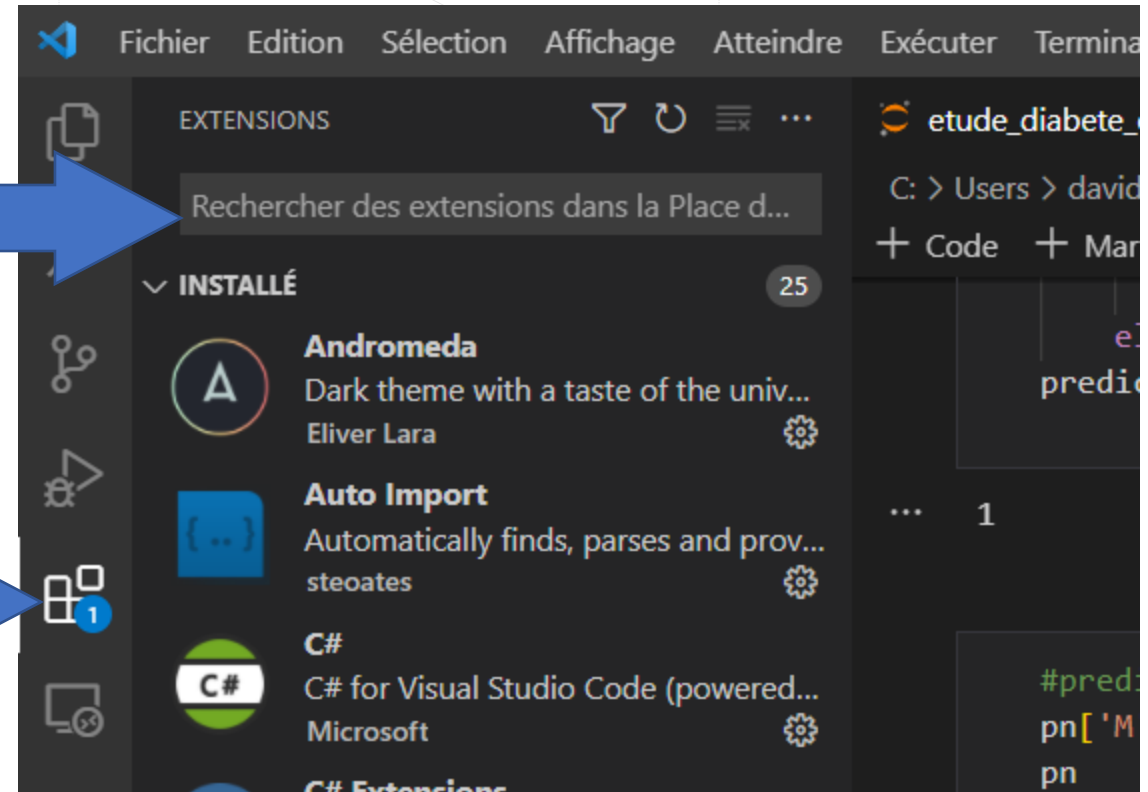


# Installer l'extention Database Client

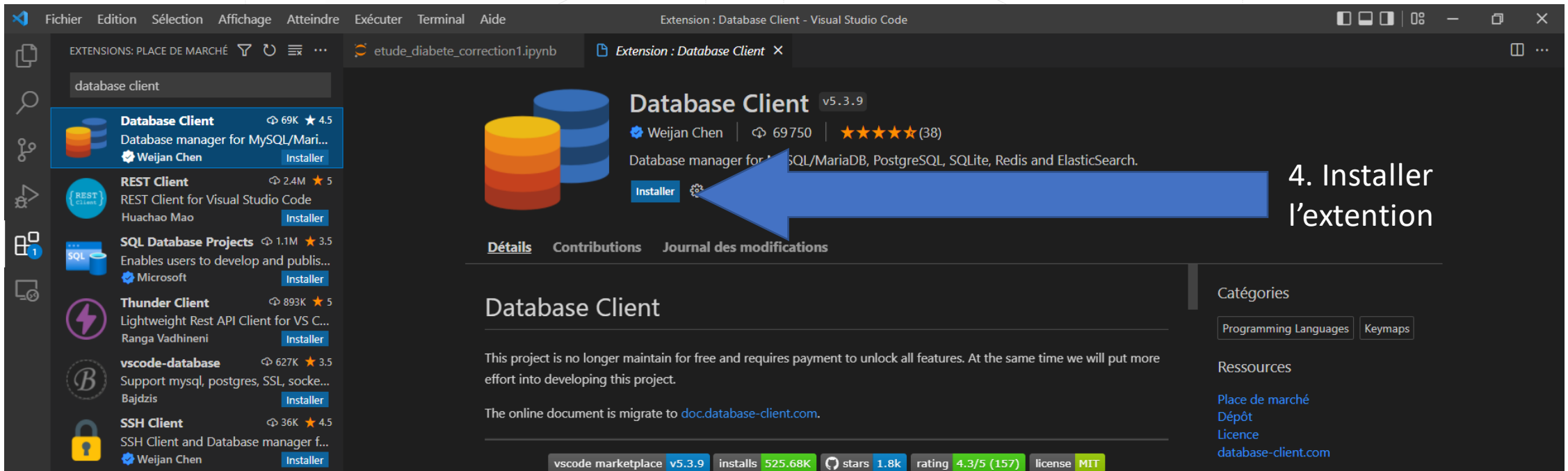
1. Ouvrir VS Code

3. Rechercher : Database Client

2. Cliquez sur le bouton Extensions



# Installer l'extension Database Client



The screenshot shows the Visual Studio Code interface with the 'Database Client' extension page. The left sidebar lists several extensions, with 'Database Client' at the top. The main panel displays the details for 'Database Client' by Weijan Chen, version 5.3.9. A large blue arrow points to the 'Installer' button. The extension description states it is a database manager for MySQL/MariaDB, PostgreSQL, SQLite, Redis, and Elasticsearch. The bottom of the page shows statistics: 525.68K installs, 1.8k stars, and a 4.3/5 rating from 157 reviews. The license is MIT.

**Database Client** v5.3.9  
 Weijan Chen | 69750 | ★★★★★ (38)  
 Database manager for MySQL/MariaDB, PostgreSQL, SQLite, Redis and Elasticsearch.

[Installer](#)

**4. Installer l'extension**

**Database Client**

This project is no longer maintain for free and requires payment to unlock all features. At the same time we will put more effort into developing this project.

The online document is migrate to [doc.database-client.com](https://doc.database-client.com).

vscode marketplace v5.3.9 installs 525.68K stars 1.8k rating 4.3/5 (157) license MIT

**Catégories**  
 Programming Languages Keymaps

**Ressources**  
[Place de marché](#)  
[Dépôt](#)  
[Licence](#)  
[database-client.com](https://database-client.com)



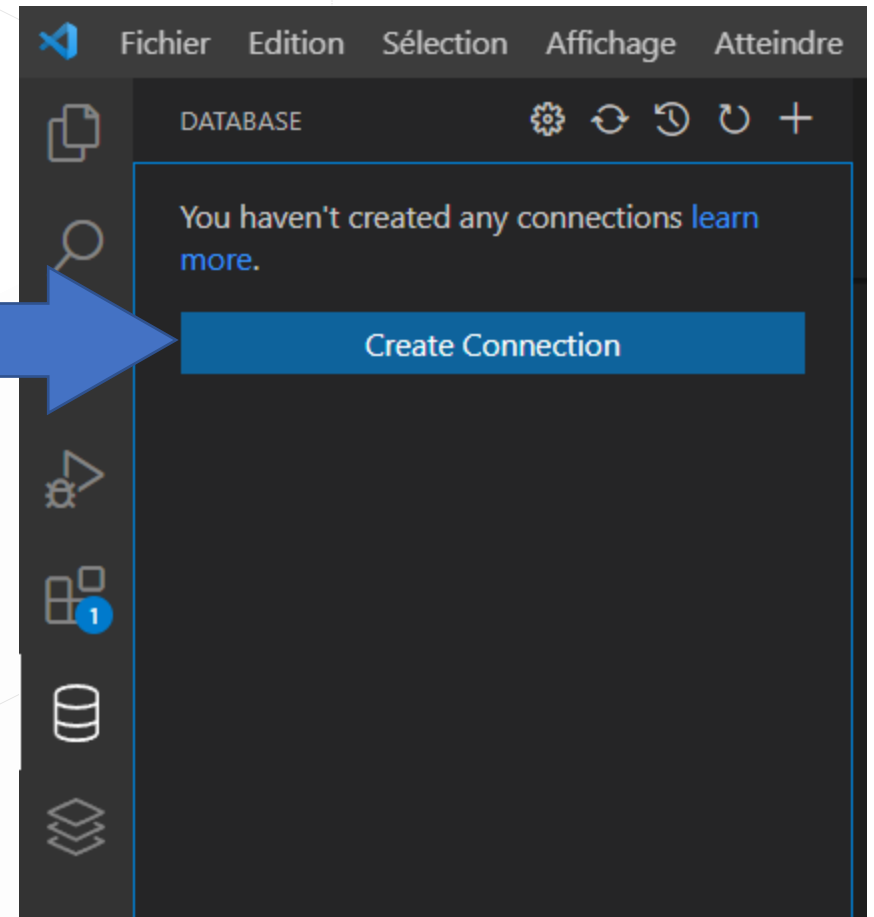
# Installer l'extension Database Client



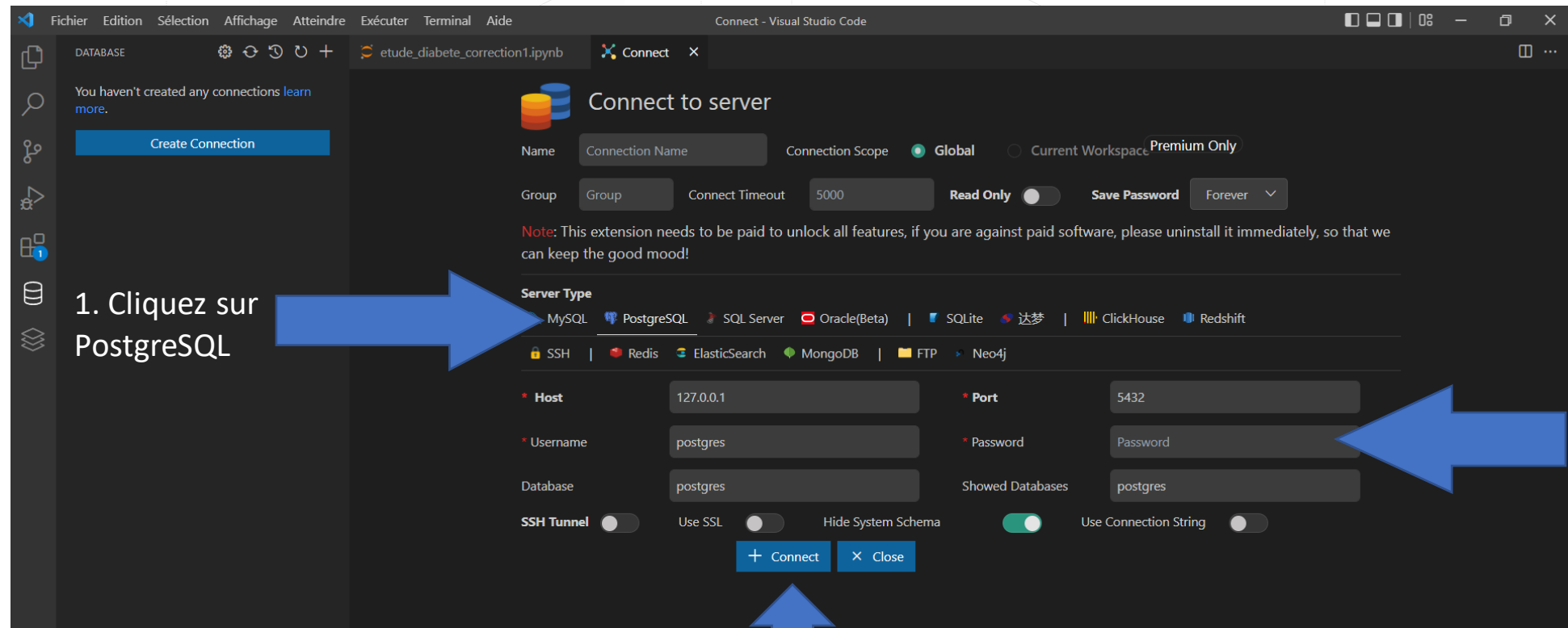
2 boutons sont  
apparus

Cliquez sur le premier  
(en forme de cylindre)

Cliquez sur  
Create Connection



# Installer l'extention Database Client

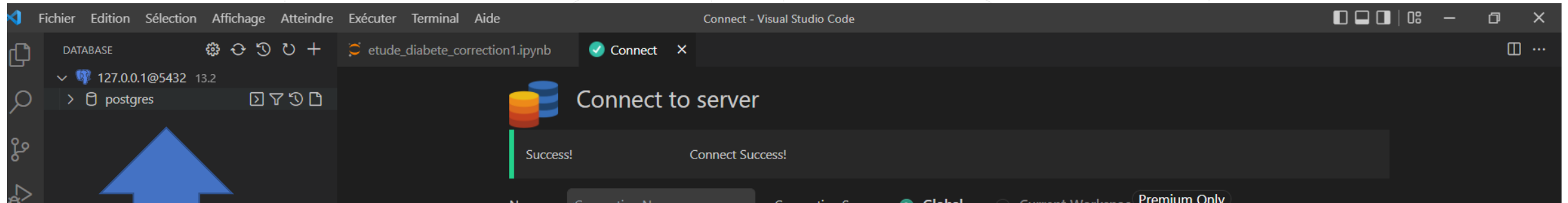


1. Cliquez sur PostgreSQL

2. Insérez votre mot de passe

3. Puis validez

# Installer l'extension Database Client



Vous avez réussi à établir une connexion à la base de données

# Sélectionner toute ou quelques colonnes

Exemples :

```
SELECT * FROM table;
```

permet de choisir toutes les colonnes d'une table

```
SELECT colonne FROM table;
```

permet de choisir une colonne d'une table

```
SELECT colonne1, colonne2 FROM table;
```

permet d'afficher 2 colonnes d'une table

```
SELECT * FROM utilisateurs;
```

```
SELECT nom FROM utilisateurs;
```

```
SELECT nom, prenom FROM utilisateurs;
```

# Sélectionner des éléments avec des conditions

Exemples :

```
SELECT * FROM table WHERE colonne1 = 'valeur1';
```

**WHERE** permet de définir une **condition**

```
SELECT * FROM table WHERE colonne1 IS NULL;
```

Permet de chercher la valeur null

```
SELECT * FROM table WHERE colonne1 IS NOT NULL;
```

Permet de chercher la valeur non null

```
SELECT * FROM utilisateurs WHERE nom = 'DUPONT';
```

```
SELECT * FROM utilisateurs WHERE voiture IS NULL;
```

(l'utilisateur ne possède pas de voiture)

```
SELECT * FROM utilisateurs WHERE voiture IS NOT NULL;
```

(l'utilisateur possède une voiture)

# Sélectionner des éléments distincts

Exemples :

`SELECT distinct colonne1 FROM table;`  
 permet de choisir des valeurs distinctes d'une table

`SELECT distinct nom FROM utilisateurs;`  
 (chaque nom sera affiché 1 fois : s'il y a deux « DUPONT », il sera présent qu'une fois dans le résultat)

ID	PRENOM	NOM
1	Alain	LAFARGE
2	Raymond	DUPONT
3	Ahmed	SLIMANI
4	Arielle	DUPONT
5	Erwan	LAFONT

NOM
LAFARGE
DUPONT
SLIMANI
LAFONT

# Le SQL : Sélectionner avec des opérateurs de comparaison

Il est possible d'utiliser les opérateurs de comparaison suivant :

Opérateur	Description
=	Égale
<>	Pas égale
!=	Pas égale
>	Supérieur à
<	Inférieur à
>=	Supérieur ou égale à
<=	Inférieur ou égale à

```
SELECT * FROM Utilisateurs WHERE age <= 40;
```

```
SELECT * FROM Utilisateurs WHERE nom != 'JOHNS';
```



# Sélectionner avec plusieurs conditions

Exemples :

```
SELECT * FROM table  
WHERE conditionA AND conditionB;
```

L'élément sélectionné doit être  
dans la condition A **et** la condition B

```
SELECT * FROM table  
WHERE conditionA OR conditionB;
```

L'élément sélectionné doit être dans  
la condition A **ou** la condition B

```
SELECT * FROM table  
WHERE NOT condition;
```

L'élément sélectionné **ne doit pas**  
être dans la condition

```
SELECT * FROM utilisateurs  
WHERE prenom = 'Jean' AND nom = 'DUPOND';
```

```
SELECT * FROM utilisateurs  
WHERE nom = 'DURAND' OR nom = 'DUPOND';
```

```
SELECT * FROM utilisateurs  
WHERE NOT nom = 'DURAND';
```

# Sélectionner avec un modèle

```
SELECT * FROM table WHERE colonne LIKE modele
```

- LIKE **'%a'** : le caractère **"%"** est un caractère joker qui remplace tous les autres caractères. Ainsi, ce modèle permet de rechercher toutes les chaînes de caractère qui se terminent par un **"a"**.
- LIKE **'a%'** : ce modèle permet de rechercher toutes les lignes de **"colonne"** qui commencent par un **"a"**.
- LIKE **'%a%'** : ce modèle est utilisé pour rechercher tous les enregistrements qui utilisent le caractère **"a"**.
- LIKE **'pa%on'** : ce modèle permet de rechercher les chaînes qui commencent par **"pa"** et qui se terminent par **"on"**, comme **"pantalon"** ou **"pardon"**.
- LIKE **'a\_c'** : peu utilisé, le caractère **"\_"** (underscore) peut être remplacé par n'importe quel caractère, mais un seul caractère uniquement (alors que le symbole pourcentage **"%"** peut être remplacé par un nombre incalculable de caractères). Ainsi, ce modèle permet de retourner les lignes **"aac"**, **"abc"** ou même **"azc"**.

# Sélectionner avec plusieurs valeurs ou une fourchette de valeurs

```
SELECT * FROM table
```

```
WHERE colonne1 IN ( valeur1, valeur2, valeur3,... );
```

Permet d'avoir les résultats qui ont une valeur parmi les valeurs qui sont dans le **IN**

```
SELECT * FROM table WHERE colonne1
```

```
BETWEEN valeur1 AND valeur2;
```

Permet d'avoir les lignes qui ont une valeur comprise la valeur 1 et la valeur 2

```
SELECT colonne1, colonne2 FROM table
```

```
ORDER BY colonne1;
```

Permet de ranger les lignes dans l'ordre de la colonne 1, ajoutez DESC à la fin pour avoir le sens inverse.

Exemples :

```
SELECT * FROM utilisateurs
```

```
WHERE nom IN ( 'DUPOND', 'DURAND', 'LAFONT' );
```

```
SELECT * FROM utilisateurs WHERE age
```

```
BETWEEN 18 AND 25;
```

```
SELECT nom, age FROM utilisateurs ORDER BY age;
```

# Sélectionner dans un ordre

```
SELECT colonne1, colonne2 FROM table
ORDER BY colonne1;
```

Permet de ranger les lignes dans l'ordre de la colonne 1, ajoutez DESC à la fin pour avoir le sens inverse.

ID	NOM	AGE
1	Serge	52
2	Ahmed	61
3	Fanny	29
4	Karima	18

Exemples :

```
SELECT prenom, age FROM utilisateurs ORDER BY age;
```

ID	PRENOM	AGE
4	Karima	18
3	Fanny	29
1	Serge	52
2	Ahmed	61

## Le SQL : Sélectionner avec des fonctions

SELECT COUNT(*) FROM table	permet de compter le nombre d'enregistrement dans une table.
SELECT AVG(colonne1) FROM table	permet de calculer une valeur moyenne sur un ensemble d'enregistrement
SELECT MIN(colonne1) FROM table	permet de retourner la plus petite valeur d'une colonne sélectionnée.
SELECT MAX(colonne1) FROM table	permet de retourner la plus grande valeur d'une colonne sélectionnée.

# Créer une table : les différents types de donnée

**Les différents types de données :**  
trois types principaux de données

- **Numérique :**
  - Pour les **Nombres entiers** on utilise souvent : **INT**
  - Pour les **Nombres décimaux** on utilise **FLOAT**
- **Caractère : VARCHAR (longueur)**
  - **VARCHAR (10)** signifie une chaîne de 10 caractères maximum
  - Si le texte est trop long (supérieur à 255 caractères) on utilise **TEXT** à la place
- **Date : DATE** : la date est stockée sous forme de **'2022-12-31'**

# Créer une table : Structure

Exemple :

```
CREATE TABLE nom_de_la_table  
(  
  colonne1 type_donnees,  
  colonne2 type_donnees,  
  colonne3 type_donnees,  
  colonne4 type_donnees  
)
```

```
CREATE TABLE utilisateurs(  
  id INT PRIMARY KEY NOT NULL,  
  nom VARCHAR(100),  
  prenom VARCHAR(100),  
  email VARCHAR(255),  
  date_naissance DATE,  
  pays VARCHAR(255),  
  ville VARCHAR(255),  
  code_postal VARCHAR(5)  
)
```



# Insérer des données

`INSERT INTO table VALUES ('valeur 1', 'valeur 2', ...)`

Si vous renseignez tous les champs

`INSERT INTO table (colonne1, colonne2, ... ) VALUES ('valeur 1', 'valeur 2', ...)`

Si vous renseignez seulement les colonnes qui vous souhaitez

Il est possible d'insérer plusieurs lignes

`INSERT INTO client (prenom, nom, ville, age)  
VALUES`

`('Rébecca', 'Armand', 'Saint-Didier-des-Bois', 24), ('Aimée', 'Hebert', 'Marigny-le-Châtel', 36),  
( 'Marielle', 'Ribeiro', 'Maillères', 27), ('Hilaire', 'Savary', 'Conie-Molitard', 58);`

Exemple :

`INSERT INTO client  
(prenom, nom, ville,  
age) VALUES  
( 'Rébecca', 'Armand',  
'Saint-Didier-des-Bois',  
24);`

# Mettre à jour des données

```
UPDATE table SET nom_colonne_1 = 'nouvelle valeur' WHERE condition;
```

*Nous modifions la colonne 1 avec une nouvelle valeur si l'enregistrement respecte la condition*

Il est possible de modifier plusieurs colonnes :

```
UPDATE table SET colonne1 = 'valeur 1', colonne2 = 'valeur 2', colonne3 = 'valeur 3'  
WHERE condition
```

```
UPDATE client SET rue = '49 Rue Ameline', ville = 'Saint-Eustache-la-Forêt', code_postal = '76210' WHERE id  
= 2;
```

# Alter Table : ajouter, renommer, modifier ou supprimer une colonne

```
ALTER TABLE nom_table add column nom_colonne type_de_donnee;
```

Permet d'ajouter une nouvelle colonne à une table déjà créée

```
ALTER TABLE nom_table rename column ancien_nom_colonne to nouveau_nom_colonne;
```

Permet de renommer une colonne déjà créée

```
ALTER TABLE nom_table modify nom_colonne type_de_donnee;
```

Permet de changer le type de donnée d'une colonne;

```
ALTER TABLE nom_table drop column nom_colonne;
```

Permet de supprimer une colonne

# Sélectionner dans 2 tables différentes

```
SELECT * FROM table_A , table_B where colonneA = colonneB;
```

colonneA a une valeur commune avec colonneB

En général, les jointures consistent à associer des lignes de 2 tables en associant l'égalité des valeurs d'une colonne d'une première table par rapport à la valeur d'une colonne d'une seconde table. Imaginons qu'une base de 2 données possède une table "utilisateurs" et une autre table "adresses" qui contient les adresses de ces utilisateurs. Avec une jointure, il est possible d'obtenir les données de l'utilisateur et de son adresse en une seule requête.

Exemple :

```
SELECT * FROM utilisateurs , adresses  
where utilisateurs.id_adresse = adresses.id ;
```

la valeur des 2 colonnes du WHERE doit etre commune

# Suppressions

**DELETE FROM** table **WHERE** condition

*Nous supprimons l'enregistrement qui respecte la condition*

Exemple :

**DELETE FROM** utilisateurs **WHERE** prenom = 'Toto';

**DROP TABLE** table;

*Nous supprimons la table intégralement*

# Documentation SQL

J'arrête ici pour les requêtes de la base de données, il n'y a pas d'autres notions intéressantes à voir, il existe des bonnes documentations, parfois avec des exemples claires :

<https://sql.sh/>  
<https://www.postgresqltutorial.com/>