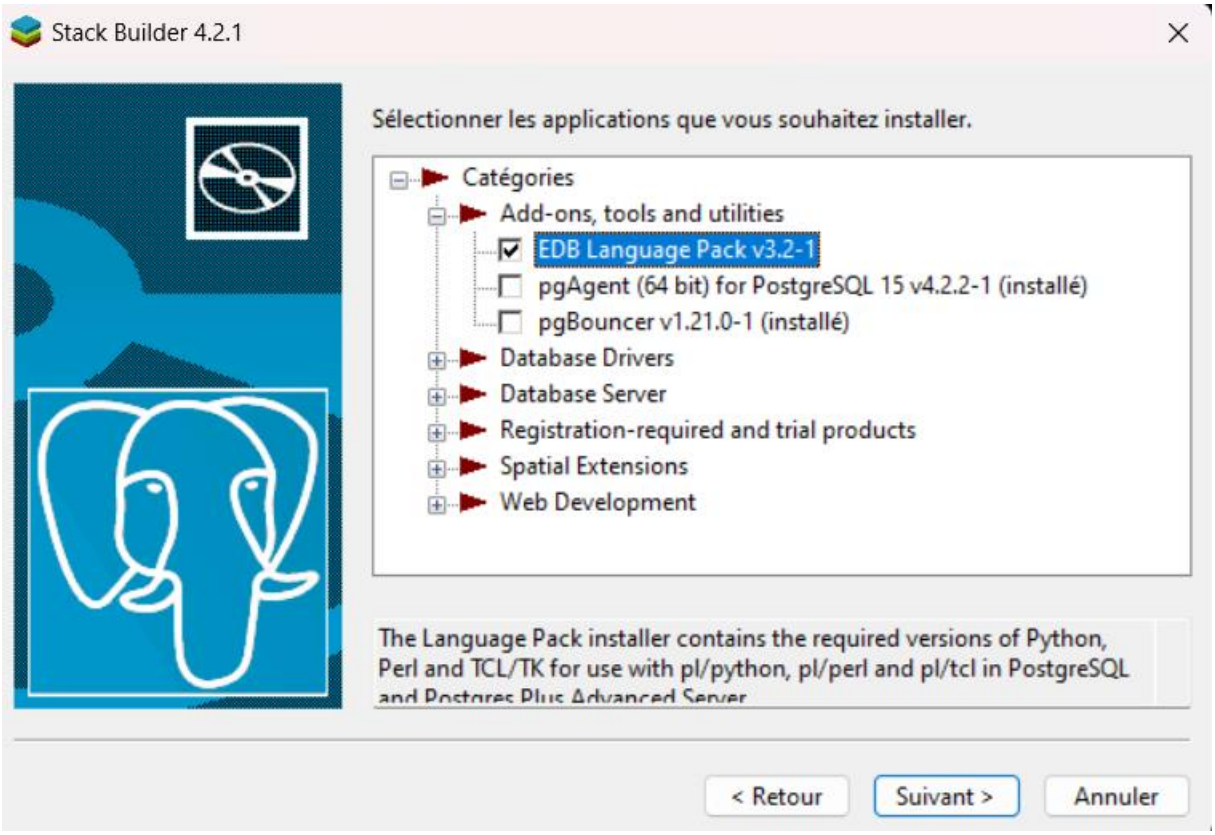
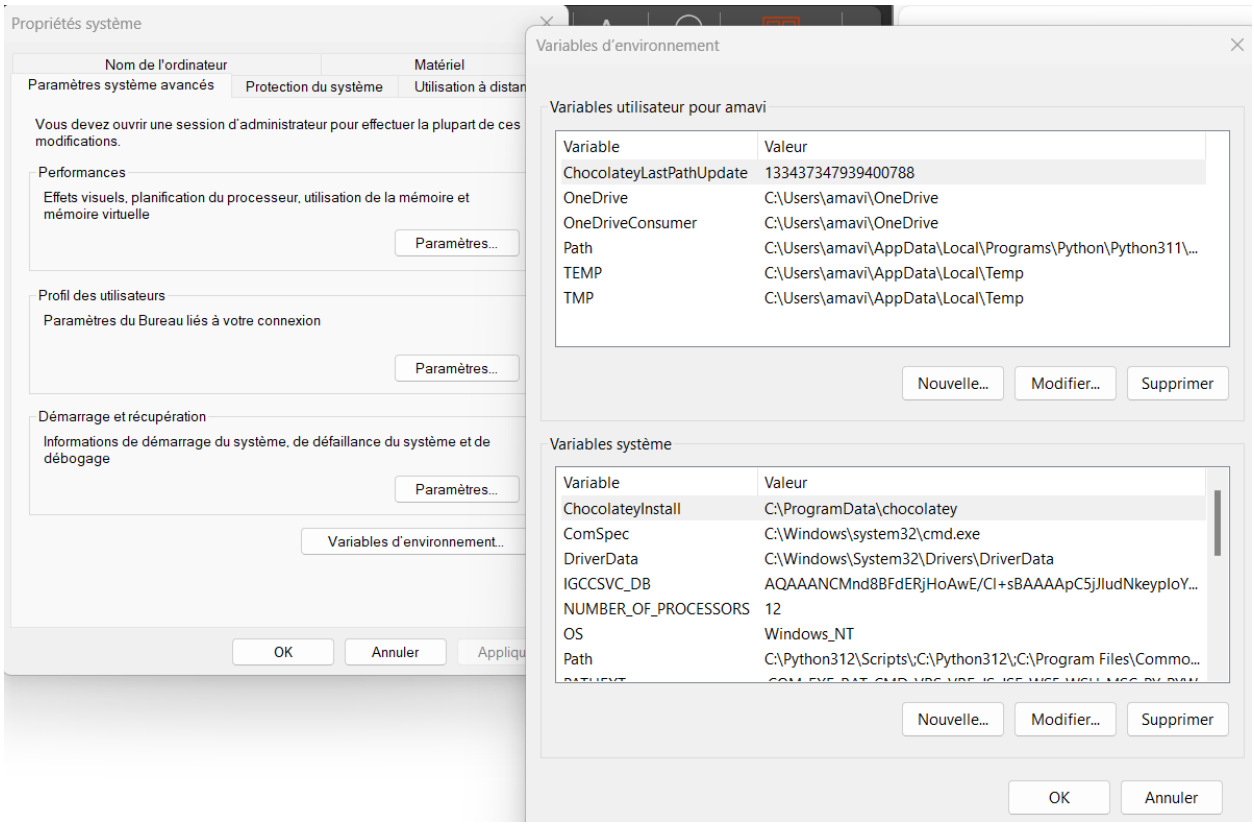
L’utilisation de Python comme langage de programmation dans le contexte de PostgreSQL.

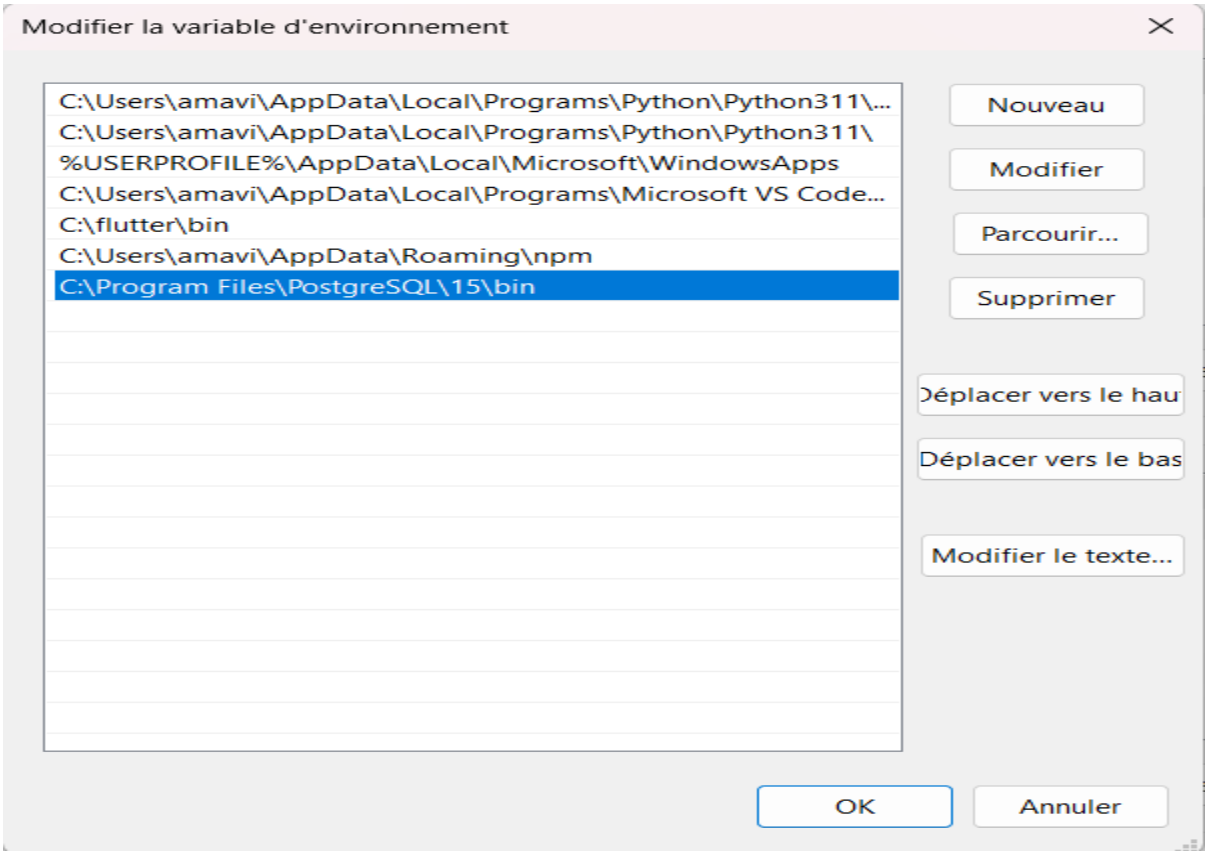
- **Installation et configuration :**
  1. **Installer PostgreSQL avec PL/Python**
     Utilisez le programme d'installation de PostgreSQL pour Windows en veillant à cocher l'option "PL/Python" lors de l'installation.



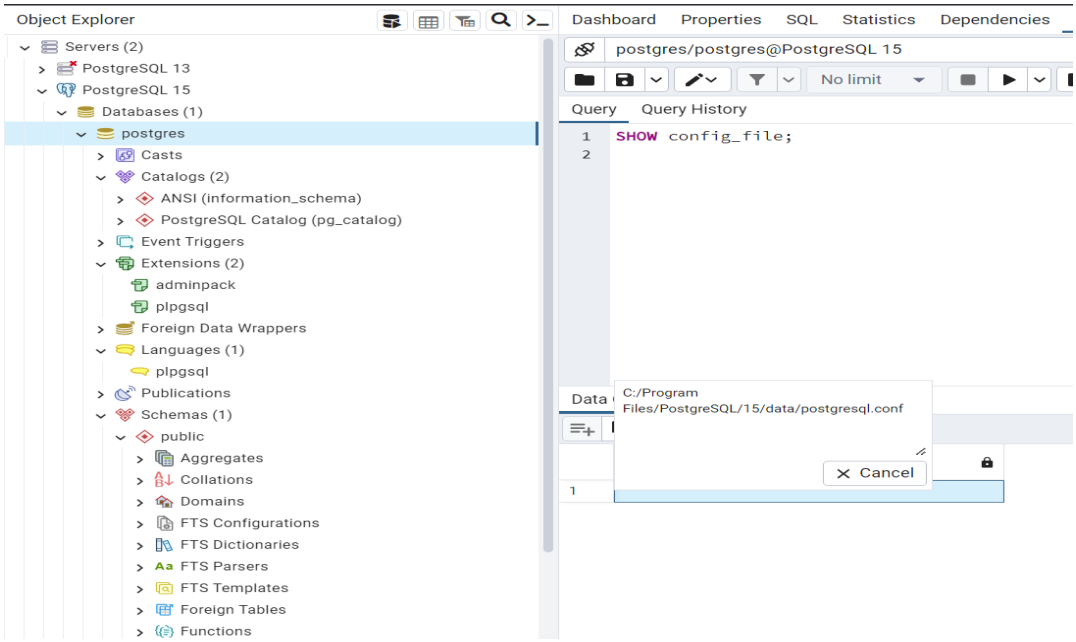
  2. **Configurer l'environnement** :

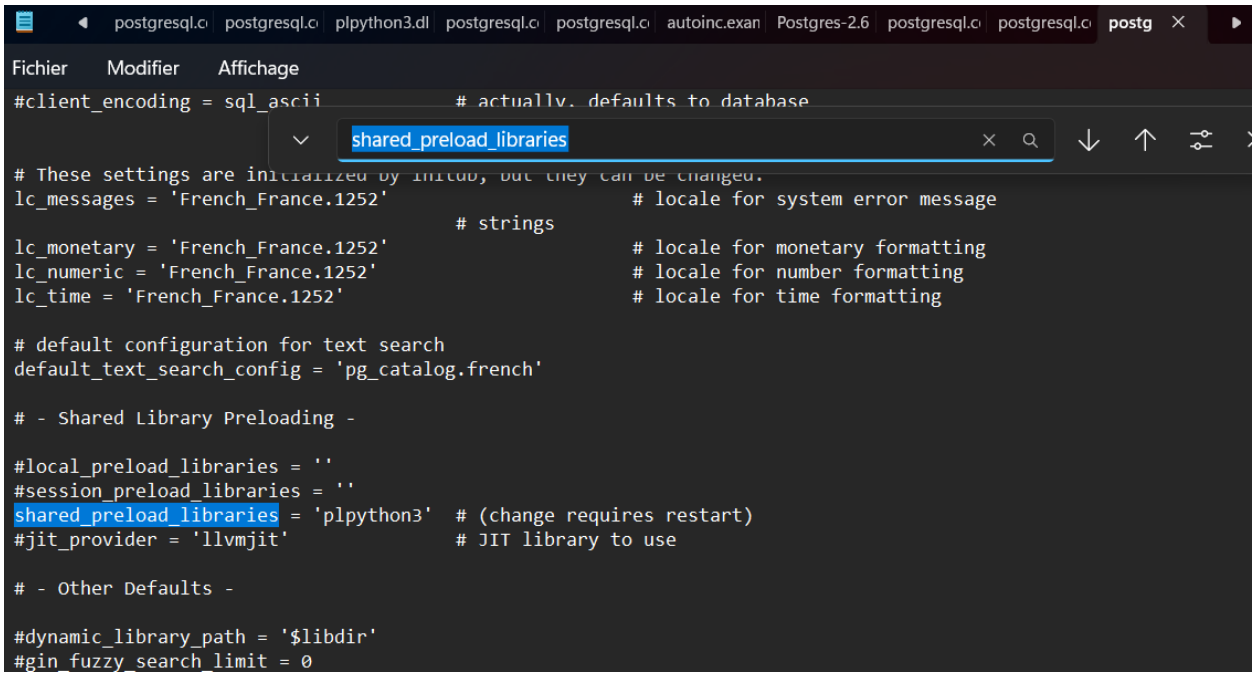
     Ajoutez le répertoire "bin" de votre installation PostgreSQL au chemin système


Propriétés système
Nom de l'ordinateur
Matériel
Paramètres système avancés
Protection du système
Utilisation à distan
Vous devez ouvrir une session d'administrateur pour effectuer la plupart de ces modifications.
Performances
Effets visuels, planification du processeur, utilisation de la mémoire et mémoire virtuelle
Paramètres...
Profil des utilisateurs
Paramètres du Bureau liés à votre connexion
Paramètres...
Démarrage et récupération
Informations de démarrage du système, de défaillance du système et de débogage
Paramètres...
Variables d'environnement...
OK
Annuler
Appliqu
Variables d'environnement
Variables utilisateur pour amavi
Variable
Valeur
ChocolateyLastPathUpdate
133437347939400788
OneDrive
C:\Users\amavi\OneDrive
OneDriveConsumer
C:\Users\amavi\OneDrive
Path
C:\Users\amavi\AppData\Local\Programs\Python\Python311\...
TEMP
C:\Users\amavi\AppData\Local\Temp
TMP
C:\Users\amavi\AppData\Local\Temp
Nouvelle...
Modifier...
Supprimer
Variables système
Variable
Valeur
ChocolateyInstall
C:\ProgramData\chocolatey
ComSpec
C:\Windows\system32\cmd.exe
DriverData
C:\Windows\System32\Drivers\DriverData
IGCCSVC_DB
AQAAANCMnd8BFdERjHoAwE/Cl+sBAAAApC5jJludNkeyploY...
NUMBER_OF_PROCESSORS
12
OS
Windows_NT
Path
C:\Python312\Scripts\;C:\Python312\;C:\Program Files\Commo...
Nouvelle...
Modifier...
Supprimer
OK
Annuler


Modifier la variable d'environnement
C:\Users\amavi\AppData\Local\Programs\Python\Python311\...
C:\Users\amavi\AppData\Local\Programs\Python\Python311\
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\Users\amavi\AppData\Local\Programs\Microsoft VS Code...
C:\flutter\bin
C:\Users\amavi\AppData\Roaming\npm
C:\Program Files\PostgreSQL\15\bin
Nouveau
Modifier
Parcourir...
Supprimer
Déplacer vers le hau
Déplacer vers le bas
Modifier le texte...
OK
Annuler

3. Activation de l'Extension PL/Python :





Redémarrage de PostgreSQL

4. Vérification de nos extensions :

```
1  SELECT * FROM pg_extension
```

Data Output | Messages | Notifications

	oid [PK] oid	extname name	extowner oid	extnamespace oid	extrelocatable boolean	extversion text	extconfig oid[]	extcondition text[]
1	13535	plpgsql	10	11	false	1.0	[null]	[null]
2	16384	adminpack	10	11	false	2.1	[null]	[null]

```
1  SELECT * FROM pg_extension
```

Data Output | Messages | Notifications

	oid [PK] oid	extname name	extowner oid	extnamespace oid	extrelocatable boolean	extversion text	extconfig oid[]	extcondition text[]
1	13535	plpgsql	10	11	false	1.0	[null]	[null]
2	16384	adminpack	10	11	false	2.1	[null]	[null]

Query | Query History

```
1  CREATE EXTENSION plpython3u
```

Data Output | Messages | Notifications

```
ERROR:  n'a pas pu charger la bibliothèque « C:/Program Files/PostgreSQL/15/lib/plpython3.dll » : The specified module could not be found.

ERREUR:  n'a pas pu charger la bibliothèque « C:/Program Files/PostgreSQL/15/lib/plpython3.dll » : The specified module could not be found.
SQL state: 58P01
```

- ## Création de Fonctions :

  Les fonctions en PL/Python sont déclarées via la syntaxe standard **CREATE FUNCTION** :

```
CREATE FUNCTION funcname (argument-list)
  RETURNS return-type
AS $$
  # PL/Python function body
$$ LANGUAGE plpython3u;
```

Par exemple, une fonction permettant de renvoyer le plus grand de deux entiers peut être définie comme :

```
CREATE FUNCTION pymax (a integer, b integer)
  RETURNS integer
AS $$
  if a > b:
    return a
  return b
$$ LANGUAGE plpython3u;
```

Le code Python fourni comme corps de la définition de fonction est transformé en fonction Python. Par exemple, ce qui precede donne:

```
def __plpython_procedure_pymax_23456():
  if a > b:
    return a
  return b
```

```
CREATE OR REPLACE FUNCTION add_numbers(a integer, b integer)
RETURNS integer AS $$
BEGIN
  RETURN a + b;
END;
$$ LANGUAGE plpython3;
```

```
CREATE OR REPLACE FUNCTION sum_list(elements integer[])
RETURNS integer AS $$
BEGIN
  RETURN SUM(elements);
END;
$$ LANGUAGE plpython3;
```

```
CREATE FUNCTION return_arr()
  RETURNS int[]
AS $$
return [1, 2, 3, 4, 5]
$$ LANGUAGE plpython3u;

SELECT return_arr();
 return_arr
-------------
 {1,2,3,4,5}
(1 row)
```

4. Accès aux données de la base de données

Créez une table employees avec des données fictives et écrivez une fonction pour obtenir le salaire moyen.

```
CREATE FUNCTION average_salary() RETURNS float
AS $$
   plan = plpy.prepare("SELECT salary FROM employees", ["float"])
   result = plpy.execute(plan)
   salaries = [x["salary"] for x in result]
   return sum(salaries) / len(salaries)
$$ LANGUAGE plpythonu;
```

- **Gestion des Erreurs dans les Fonctions PL/Python**

  Utilisation de blocs try-except :

  Les blocs try-except peuvent être utilisés pour gérer les erreurs.
  Exemple :

```
CREATE OR REPLACE FUNCTION divide(a integer, b integer)
RETURNS float AS $$
BEGIN
  BEGIN
    RETURN a::float / b;
  EXCEPTION
    WHEN ZERO_DIVIDE THEN
      RETURN NULL; -- Gestion de la division par zéro
  END;
END;
$$ LANGUAGE plpython3;
```

- **Optimisation des Performances :**
  - **Conseils pour optimiser les performances :**
    - Évitez les boucles Python lorsque des opérations de jeu de données peuvent être effectuées en SQL.
    - Utilisez des types de données natifs PostgreSQL plutôt que des types Python pour améliorer l'efficacité.
  - **Bonnes pratiques pour éviter les pièges courants :**
    - Minimisez l'utilisation des opérations I/O coûteuses dans les fonctions PL/Python.
    - Indexez correctement les tables utilisées dans les fonctions.

- **Exemples d'Utilisation Réelle :**
  - **Cas d'utilisation réelle :**
    - Un système de recommandation utilisant des algorithmes complexes écrits en Python pour analyser les préférences des utilisateurs.

Amélioration de la logique métier et gestion des données :

- PL/Python permet d'intégrer des analyses avancées directement dans la base de données, facilitant ainsi la gestion des données sans déplacement externe.