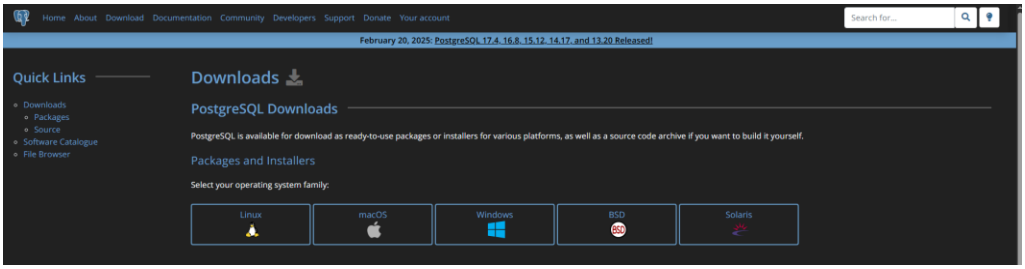
TP3 : PL/PGSQL – Partie 1

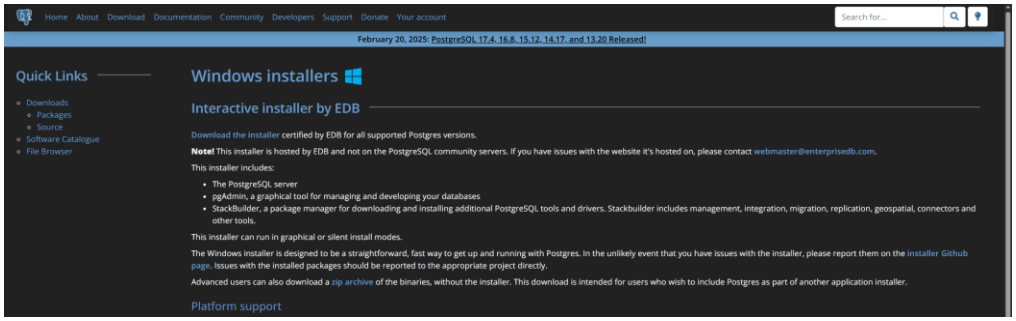
1. Installation de PostgreSQL(Windows)

- Télécharger PostgreSQL Rendez-vous sur le site officiel de PostgreSQL : https://www.postgresql.org/

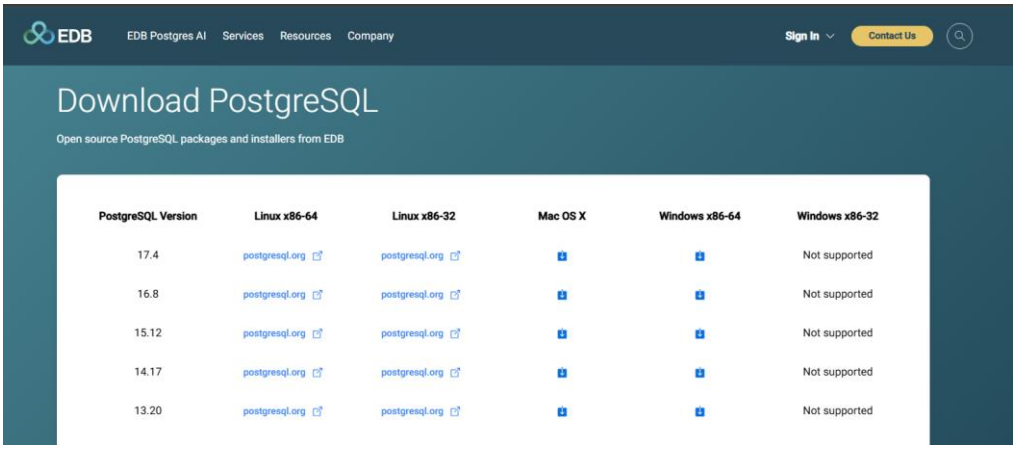
- Cliquez sur **"Download"**, puis sélectionnez la version adéquate à votre SE.



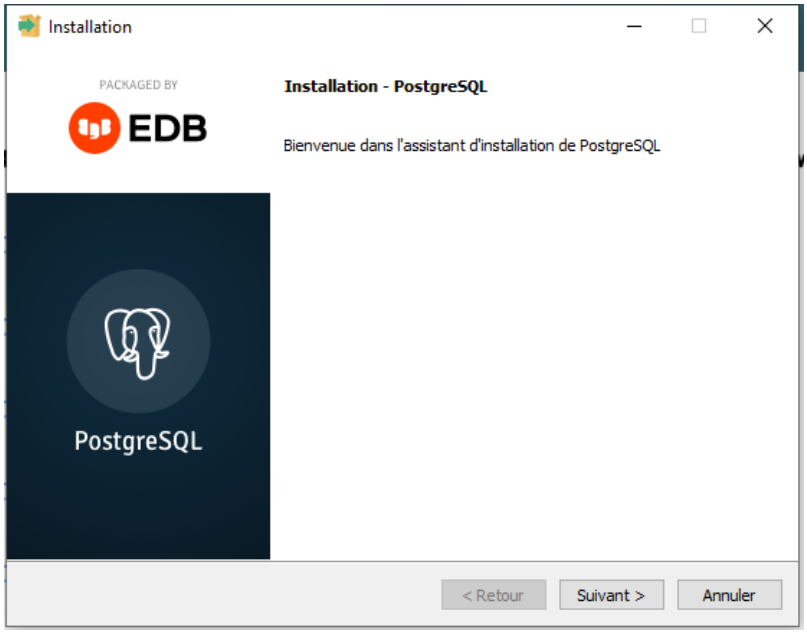
- Sur windows, vous pouvez effectuer l'installation avec l'utilitaire graphique



- Téléchargez la dernière version stable (actuellement PostgreSQL 17).

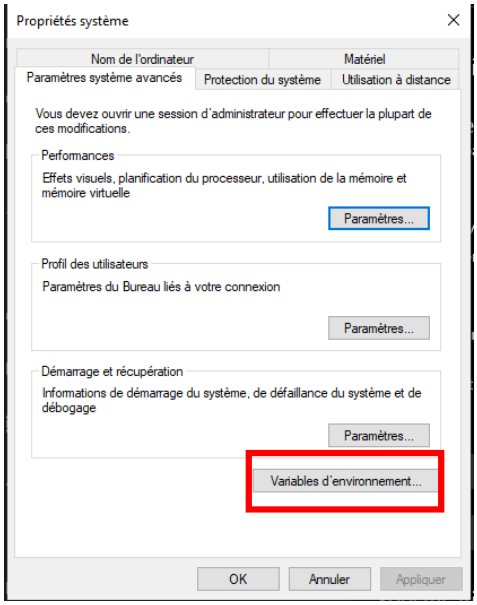


- Ouvrez le fichier d'installation et suivez l'assistant d'installation.

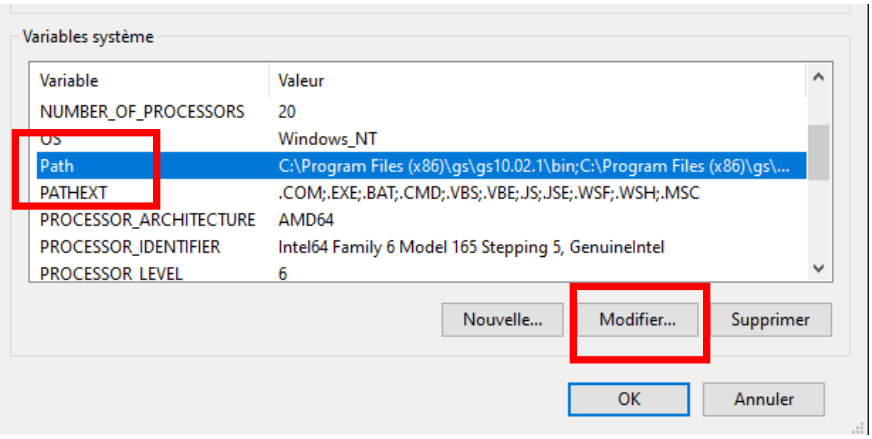


- Configurer PostgreSQL pendant l'installation
  - Définissez un mot de passe pour l'utilisateur postgres.
  - Sélectionnez le port par défaut 5432 (ou un autre si nécessaire).
  - Finalisez l'installation et attendez que PostgreSQL soit correctement installé.

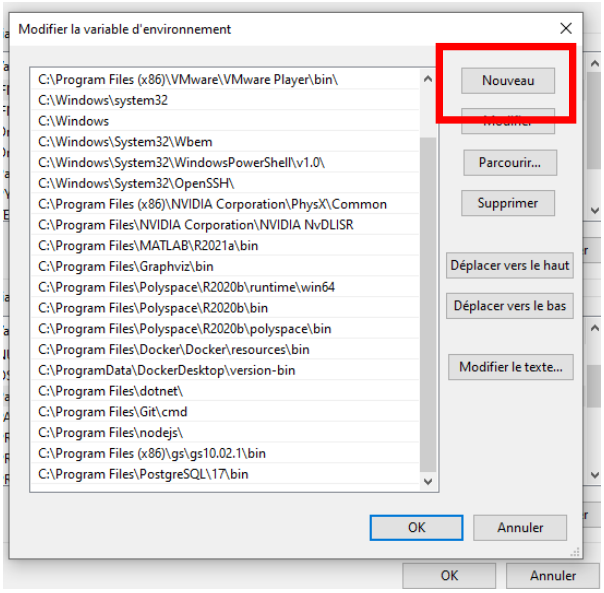
- Ajouter PostgreSQL à la variable d'environnement PATH. Pour exécuter psql depuis l'invite de commandes, ajoutez PostgreSQL au PATH :
    - Ouvrez Paramètres système avancés.
    - Cliquez sur Variables d'environnement.



    - Dans la section Variables système, sélectionnez Path, puis cliquez sur Modifier.



    - Cliquez sur Nouveau et ajoutez le chemin suivant (à adapter selon votre version) :



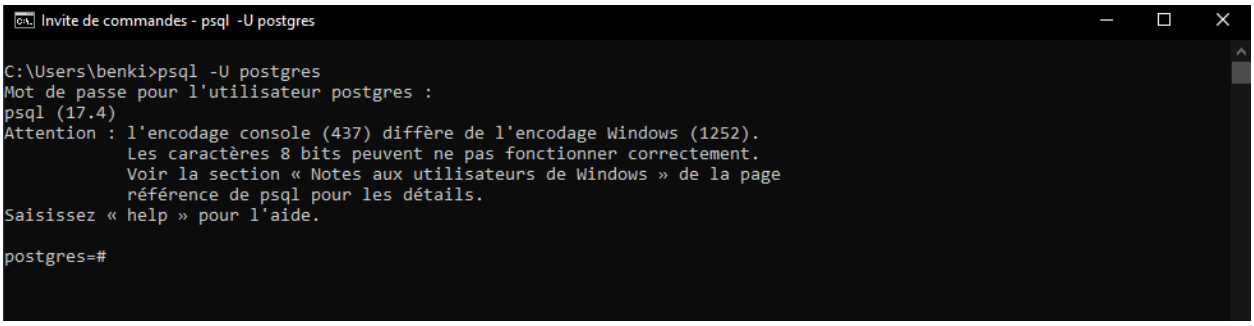
C:\Program Files\PostgreSQL\17\bin

2. Vérifier l'installation

- Ouvrez l'invite de commandes (`cmd`) et testez PostgreSQL avec :

psql -U postgres

- Indiquer votre mot de passe (défini lors de l'installation). Si l'installation est réussie, vous devriez voir l'invite `postgres=#` indiquant que vous êtes connecté.



3. Quelques commandes essentielles

- **Psql** :c'est l'outil en ligne de commande de PostgreSQL, utilisé pour exécuter des requêtes SQL et interagir avec la base de données.
- **-U postgres :**
    - **-U** signifie **User** (utilisateur).
    - **postgres** est le nom de l'utilisateur avec lequel vous souhaitez vous connecter.

Lorsque vous exécuter la commande suivante :

```
psql -U postgres
```

Par défaut, PostgreSQL installe un utilisateur administrateur appelé ***postgres***. Cette commande essaie de se connecter avec cet utilisateur à la base de données du même nom (postgres si aucune base n'est spécifiée).

- Pour se connecter à une base de données spécifique :

```
psql -U postgres -d ma_base
```

- Pour se connecter via un hôte spécifique (ex. : serveur distant ou local) :

```
psql -U postgres -h localhost
```

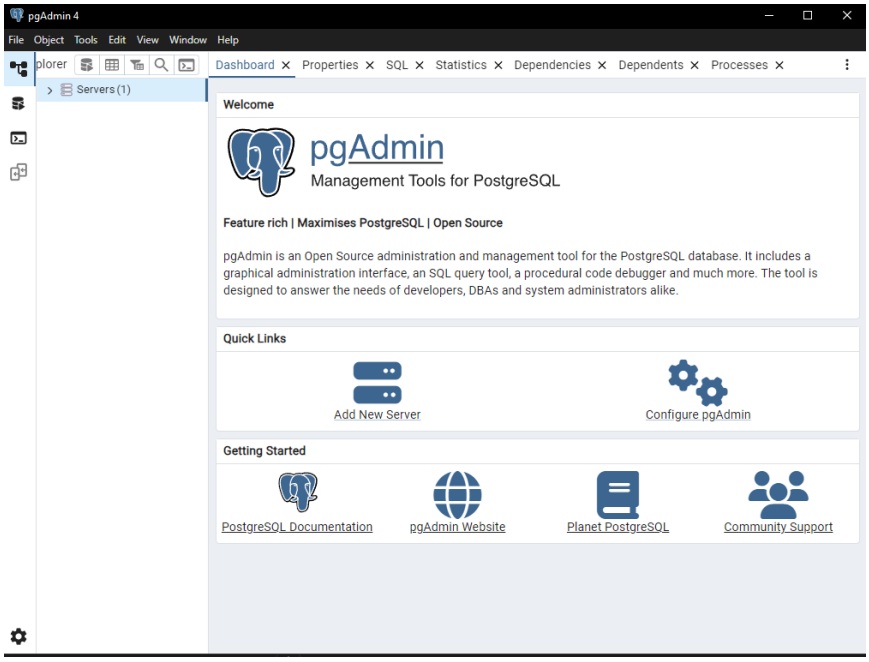
- Pour se connecter via un port spécifique

```
psql -U postgres -h localhost -p 5433
```

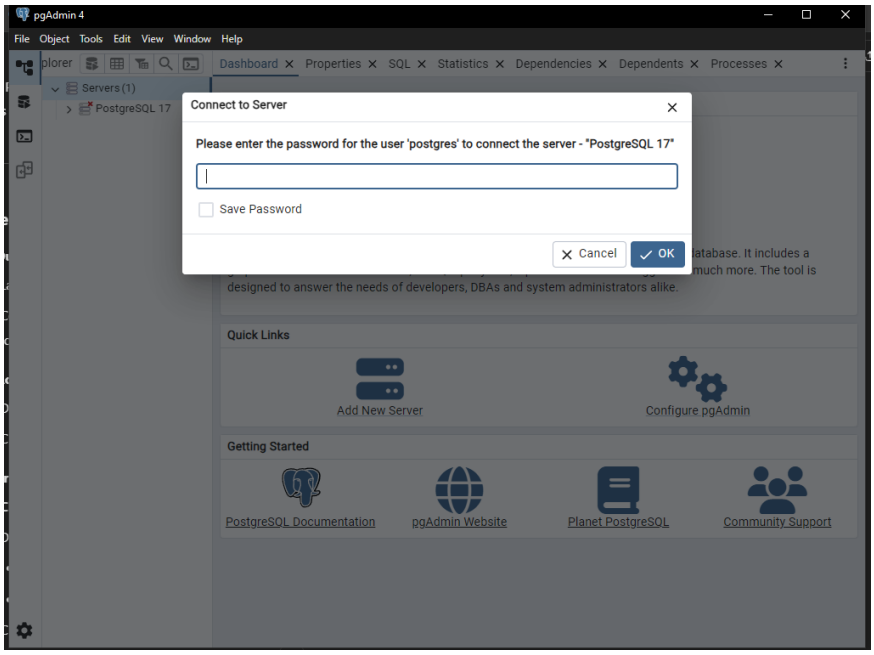
4. pgAdmin4

pgAdmin 4 est l'outil graphique pour PostgreSQL, similaire à **phpMyAdmin** pour MySQL, qui permet de créer des bases de données, des tables et d'exécuter des requêtes SQL, fonctions et procédures de manière intuitive sans avoir à passer par la ligne de commande. Pour y accéder :

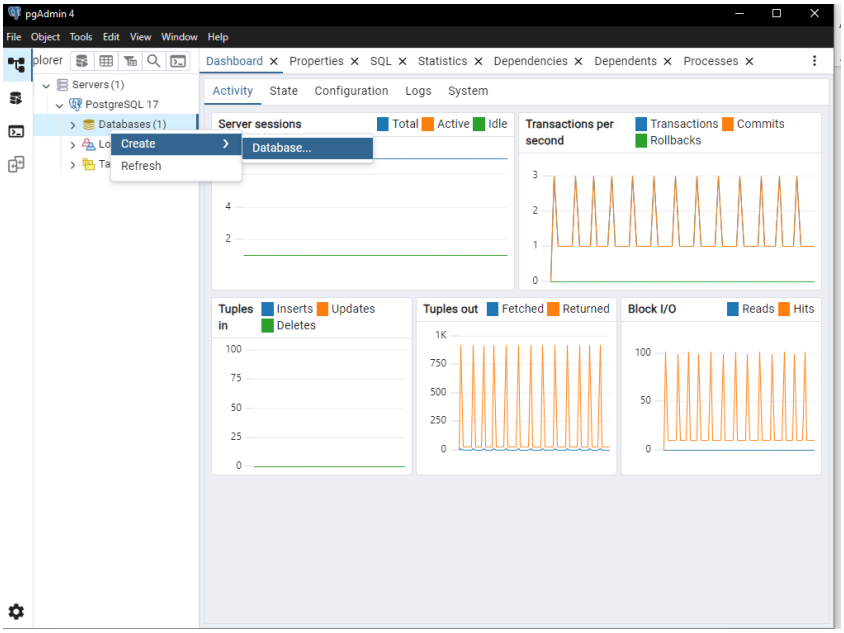
- Ouvrez le dossier où PostgreSQL est installé.
- Allez dans le dossier pgAdmin 4.
- Lancez l'application pgAdmin 4.



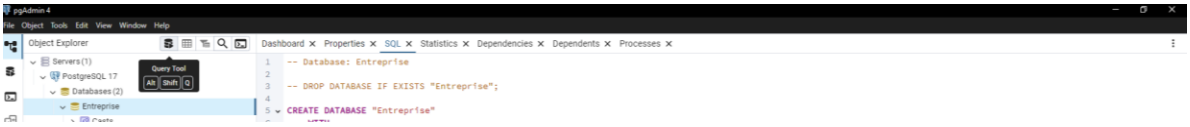
- Une fois ouvert, vous pouvez vous connecter à votre serveur PostgreSQL, en insérant votre mdp.



- Pour créer une nouvelle base de données, clic droit sur Database ➔ Create.



- Pour exécuter des requêtes SQL, des fonctions PL/PGSQL , cliquez sur Query Tool dans le menu en haut



Exercice 1

Objectif : Dans cet exercice, vous allez créer une base de données d'entreprise, définir les tables nécessaires, insérer des enregistrements, puis appliquer des concepts avancés de SQL et PL/pgSQL, notamment les blocs anonymes, les blocs nommés, les fonctions et les variables.

Création de la base de données avec pgAdmin

- Ouvrez pgAdmin 4 et connectez-vous à votre serveur PostgreSQL.
- Créez une nouvelle base de données appelée entreprise.

Création des tables

Dans la base de données entreprise, vous allez créer les tables suivantes :

- Table des employés (employes) :
  - id (type SERIAL, clé primaire)
  - nom (type VARCHAR)

    - prenom (type VARCHAR)
    - poste (type VARCHAR)
    - salaire (type DECIMAL)
    - date_embauche (type DATE)
    - id_departement (types INTEGER),
- Table des départements (departements) :
    - id (type SERIAL, clé primaire)
    - nom_departement (type VARCHAR)
- Table des projets (projets) :
    - id (type SERIAL, clé primaire)
    - nom_projet (type VARCHAR)
    - date_debut (type DATE)
    - date_fin (type DATE)
    - Budget(TYPE decimal)
- Table employe_projet :
    - Emp_id (INTEGER), FK
    - Projet_ID (INTEGER), FK

Insertion des données :

Ajoutez au moins 3 enregistrements dans chacune des tables employes, departements et projets. Utilisez la commande INSERT INTO pour insérer les données manuellement.

Contrôle et ajustement automatique du salaire des employés

Votre entreprise souhaite s'assurer qu'aucun employé ne soit payé en dessous d'un seuil minimum.

- Créez une fonction nommée qui prend en entrée l'identifiant d'un employé.
- Cette fonction doit vérifier le salaire de l'employé et appliquer une augmentation automatique de 10% si le salaire est inférieur à 2500.
- La fonction doit retourner le salaire final après ajustement.
- Testez la fonction avec différents employés.

Simulation de primes de performance pour les employés

La direction souhaite mettre en place un système de primes basé sur la performance des employés et la rentabilité des projets.

- Créer une fonction nommée qui calcule et attribue une prime aux employés en fonction de plusieurs critères internes.

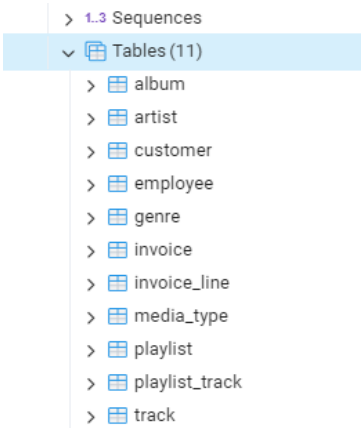
- Définir une variable globale prime initialisée à 1000, représentant une prime de base pour tous les employés.
- Dans un premier bloc interne, ajuster la valeur de prime pour les employés ayant un salaire supérieur à 5000, en la fixant à 2000.
- Dans un deuxième bloc encore plus interne, si l'employé travaille sur un projet dont le budget dépasse 100 000, alors la prime passe à 3000 pour refléter la rentabilité du projet.
- Affichez à chaque niveau les valeurs de la prime pour observer les effets du shadowing des variables.
- À la fin, la fonction doit retourner la valeur de la prime globale pour vérifier qu'elle n'a pas été modifiée par les blocs internes.
- Testez la fonction sur différents employés et projets pour voir l'impact du système de primes.

Exercice 2

Objectif : Vous allez importer la base de données Chinook dans pgAdmin (Utilisée dans le TP1) et automatiser certaines tâches courantes en utilisant SQL et PL/pgSQL.

Importez la base de données Chinook

- Ouvrez pgAdmin 4 et connectez-vous à votre serveur PostgreSQL.
- Créez une nouvelle base de données appelée Chinook.
- Ouvrez l'éditeur SQL et exécutez le fichier Chinook_PostgreSql.sql pour créer les tables et insérer les données (Fichier en PJ du sujet de TP).
- Vérifiez la structure de la BD pour s'assurer que l'importation s'est bien passé. Pour visualiser les tables importées, cliquer sur la BD chinook sur le default workspace, schémas,public,tables vous verrez une structure qui ressemble à ceci :



Tâche 1 : Calculer le total des ventes d'un artiste

- Créez une fonction qui prend en entrée l'ID d'un artiste.
- Elle doit calculer le total des ventes (montant total des factures des morceaux appartenant à cet artiste).
- Affichez un message indiquant le montant total des ventes de l'artiste.
- Tester la fonction

Tâche 2 : Augmenter automatiquement les prix des morceaux les plus vendus

- Identifiez les 10 morceaux les plus vendus (en quantité).
- Augmentez leur prix de 10% automatiquement.
- Affichez les prix avant et après mise à jour.
- **Quelques directives :**
  - Déclaration de la boucle FOR : La boucle FOR est utilisée pour itérer sur une séquence de valeurs. Vous pouvez itérer sur un ensemble de résultats de requête. Syntaxe :

```
FOR variable IN
 SELECT colonne1, colonne2 FROM table WHERE condition
LOOP
  -- Instructions à répéter
END LOOP;
```

Tâche 3 : Générer une facture pour un client donné

- Créez une fonction qui prend en paramètre un CustomerId.
- Elle doit afficher en sortie les achats du client en mentionnant l'id de la facture, la date de la fature et le montant total de sa facture.
- **Quelques directives :**
  - Rappel : Clause « INTO » : Utilisé pour récupérer des valeurs uniques dans des variables locales. **Exemple :** Si vous voulez récupérer une seule valeur (par exemple, un montant total) d'une requête et l'affecter à une variable, vous utilisez INTO. Syntaxe : `SELECT column_name INTO variable_name FROM table WHERE condition;`
  - "RETURN QUERY" : Utilisé pour retourner plusieurs lignes ou un jeu de résultats (comme une table) dans une fonction PL/pgSQL qui retourne une

table. Exemple : Dans le cas où o nsouhaite retourner les achats du clients comme ue table incluant l'id, la date et le montant total, vous devez utiliser RETURN QUERY. Syntaxe :

```
RETURN QUERY

SELECT column1, column2 FROM table WHERE condition;
```