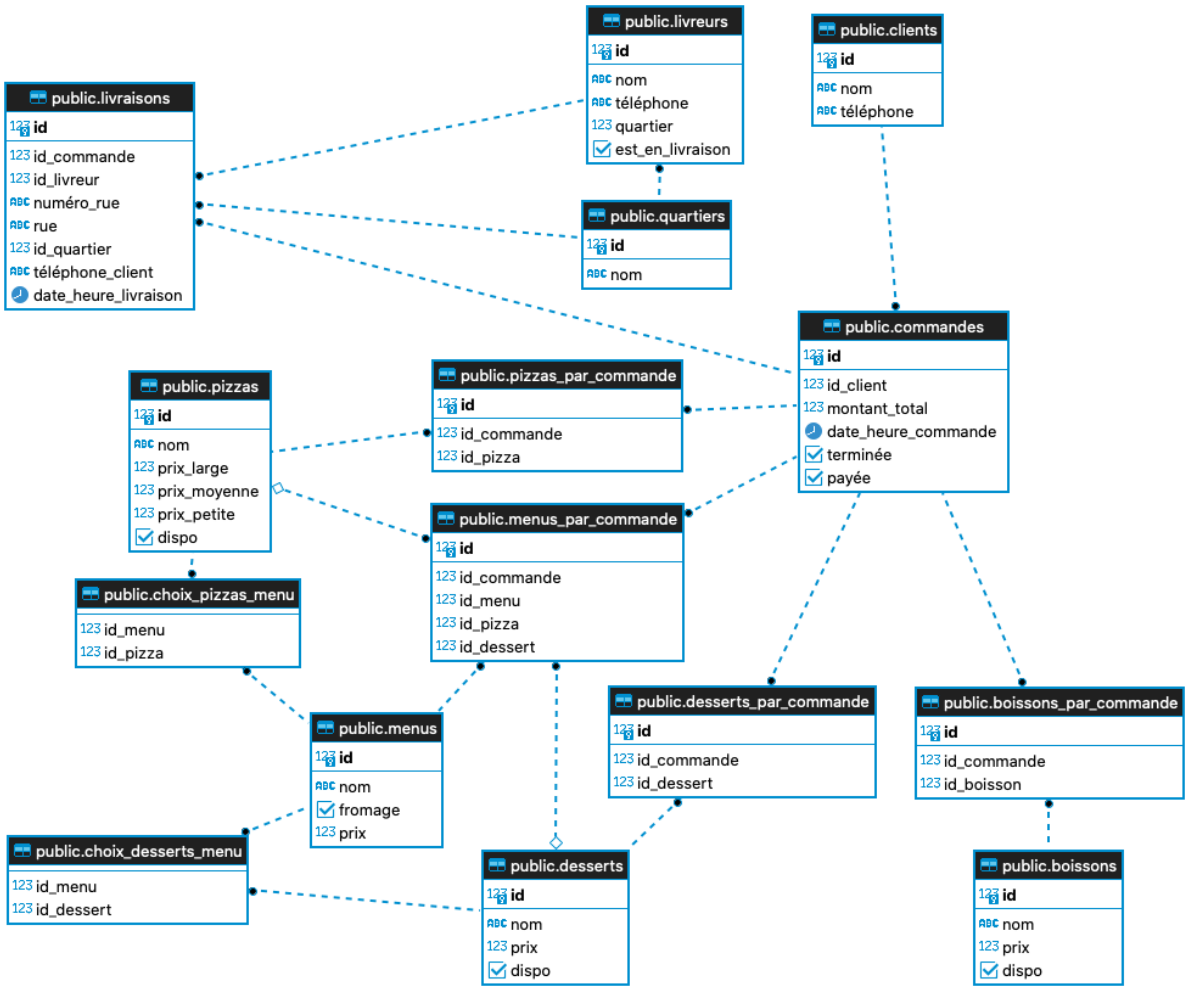
TP NFP 107

Par Jeremy et Marc

https://github.com/d4wae89d498/NFP107-TP




```sql
SELECT * from public.commander(
    nom_client => 'John Doe',
    -- LES MENUS ET LES OPTIONS DES MENUS
    liste_menus => array['Hiver', 'Eté'],
    liste_desserts_menus => array['Banane au whisky', 'Glace'],
    liste_pizzas_menus => array ['4 fromages', 'Merguez'],
    -- LES EXTRAS
    liste_pizzas => array['Raclette'],
    liste_tailles_pizzas => array['petite'],
    liste_desserts => array['Tiramisu'],
    liste_boissons => array['1664', '1664', '1664', 'Sprite'],
    -- DETAILS COMMANDE
    à_emporter => true,
    téléphone_client => '07 88 09 12 35',
    numéro_rue => '35B',
    rue => 'Rue Paul Santy',
    quartier => 'Mermoz'
)
```

	id_commande integer	id_livreur integer
1	4	7

Messages

Successfully run.
1 rows affected.

TABLE OF CONTENTS

TECHNOLOGIES 3

AUTOMATISATIONS 3

CREATION DES TABLES 4

JEU DE DONNEES : LES QUARTIERS LYONNAIS ET DES MENUS VARIES 7

LA FONCTION COMMANDER 9

AUTRES REQUETES 13

NOTES 16

TECHNOLOGIES

Nous avons choisi PostgreSQL car c'est système de gestion de base de données de haute performance disponible en source ouverte que nous souhaitons maitriser davantage. Le diagramme a été créé à l'aide de DBeaver. Nous avons aussi utilisé PgAdmin4 pour visualiser nos requêtes.

AUTOMATISATIONS

Nous avons automatisé une partie du processus de livraison avec une fonction « commander » qui prends en paramètre le nom du client, ce qui est commandé en menu ou en extra, et éventuellement des coordonnées de livraison. Notre fonction retourne un identifiant de commande et l'identifiant du livreur.

Si la commande est à livrer : notre fonction vérifie qu'un livreur est disponible pour le quartier demandé, si oui un livreur est automatiquement assigné, sinon une exception est levée. Notre fonction de commande vérifie également la cohérence des données afin de donner des messages d'erreurs clairs. De plus, dans le cadre des pizzas hors-menu ; le client peut choisir plusieurs tailles (petite, moyenne, large). Les pizzas en menus sont toutes des pizzas larges.

CREATION DES TABLES

```sql
-- Créer les tables

CREATE TABLE pizzas (
  id SERIAL PRIMARY KEY,
  nom VARCHAR(255) NOT NULL UNIQUE,
  prix_large NUMERIC(5,2) NOT NULL,
  prix_moyenne NUMERIC(5,2) NOT NULL,
  prix_petite NUMERIC(5,2) NOT NULL,
  dispo BOOLEAN NOT NULL DEFAULT(true)
);

CREATE TABLE menus (
  id SERIAL PRIMARY KEY,
  nom VARCHAR(255) NOT NULL UNIQUE,
  fromage BOOLEAN NOT NULL,
  prix DECIMAL(5,2) NOT NULL
);

CREATE TABLE choix_pizzas_menu (
  id_menu INTEGER NOT NULL REFERENCES menus (id),
  id_pizza INTEGER NOT NULL REFERENCES pizzas (id)
);

CREATE TABLE desserts (
  id SERIAL PRIMARY KEY,
  nom VARCHAR(255) NOT NULL UNIQUE,
  prix DECIMAL(5,2) NOT NULL,
  dispo BOOLEAN NOT NULL DEFAULT(true)
);

CREATE TABLE choix_desserts_menu (
  id_menu INTEGER NOT NULL REFERENCES menus (id),
  id_dessert INTEGER NOT NULL REFERENCES desserts (id)
);

CREATE TABLE clients (
  id SERIAL PRIMARY KEY,
  nom VARCHAR(255) NOT NULL UNIQUE,
  téléphone VARCHAR(15) NULL
);

CREATE TABLE commandes (
  id SERIAL PRIMARY KEY,
  id_client INTEGER NOT NULL REFERENCES clients (id),
  montant_total DECIMAL(10,2) NOT NULL,
  date_heure_commande TIMESTAMP NOT NULL default(NOW()),
  terminée BOOLEAN NOT NULL DEFAULT(false),
  payée BOOLEAN NOT NULL DEFAULT(false)
);
```

```sql
CREATE TABLE menus_par_commande (
  id SERIAL PRIMARY KEY,
  id_commande INTEGER NOT NULL REFERENCES commandes (id),
  id_menu INTEGER NOT NULL REFERENCES menus (id),
  id_pizza INTEGER NULL REFERENCES pizzas (id),
  id_dessert INTEGER NULL REFERENCES desserts (id)
);

CREATE TABLE pizzas_par_commande (
  id SERIAL PRIMARY KEY,
  id_commande INTEGER NOT NULL REFERENCES commandes (id),
  id_pizza INTEGER NOT NULL REFERENCES pizzas (id),
  taille VARCHAR(255) NOT NULL
);

CREATE TABLE desserts_par_commande (
  id SERIAL PRIMARY KEY,
  id_commande INTEGER NOT NULL REFERENCES commandes (id),
  id_dessert INTEGER NOT NULL REFERENCES desserts (id)
);

CREATE TABLE boissons (
  id SERIAL PRIMARY KEY,
  nom VARCHAR(255) NOT NULL UNIQUE,
  prix DECIMAL(5,2) NOT NULL,
  dispo BOOLEAN NOT NULL DEFAULT(true)
);

CREATE TABLE boissons_par_commande (
  id SERIAL PRIMARY KEY,
  id_commande INTEGER NOT NULL REFERENCES commandes (id),
  id_boisson INTEGER NOT NULL REFERENCES boissons (id)
);

CREATE TABLE quartiers (
   id SERIAL PRIMARY KEY,
   nom VARCHAR (255) NOT NULL UNIQUE
);

CREATE TABLE livreurs (
 id SERIAL PRIMARY KEY,
 nom VARCHAR (255) NOT NULL UNIQUE,
 téléphone VARCHAR (255) NOT NULL,
 quartier INTEGER NOT NULL REFERENCES quartiers (id),
 est_en_livraison BOOLEAN NOT NULL default (false)
);
```

```sql
CREATE TABLE livraisons (
  id SERIAL PRIMARY KEY,
  id_commande INTEGER NOT NULL REFERENCES commandes (id),
  id_livreur INTEGER NOT NULL REFERENCES livreurs (id),
  numéro_rue VARCHAR(255) NOT NULL,
  rue VARCHAR(255) NOT NULL,
  id_quartier INTEGER NOT NULL REFERENCES quartiers (id),
  téléphone_client VARCHAR (255) NOT NULL,
  date_heure_livraison TIMESTAMP NULL
);
```

JEU DE DONNEES : LES QUARTIERS LYONNAIS ET DES MENUS VARIES

```sql
-- Transaction pour insérer les données

BEGIN TRANSACTION;

INSERT INTO pizzas (nom, prix_large, prix_moyenne, prix_petite, dispo)
VALUES
('4 fromages', 12.50, 9.50, 7.50, true),
('Margherita', 10.50, 8.50, 6.50, true),
('Pepperoni', 11.50, 9.50, 7.50, true),
('Végétarienne', 11.00, 8.50, 6.50, true),
('BBQ New York', 12.50, 9.50, 7.50, true),
('Merguez', 13.50, 10.50, 8.50, true),
('Raclette', 12.00, 9.00, 7.00, true);

INSERT INTO boissons (nom, prix, dispo)
VALUES
('Coca', 2.00, true),
('1664', 3.00, true),
('Sprite', 2.00, true),
('Ice Tea', 2.00, true);

INSERT INTO desserts (nom, prix, dispo)
VALUES
('Tiramisu', 3.50, true),
('Panna cotta', 3.50, true),
('Mousse au chocolat', 3.50, true),
('Glace', 3.50, true),
('Macarons crème', 4.00, true),
('Banane au whisky', 12.00, true);

INSERT INTO menus (nom, fromage, prix)
VALUES
       ('Printemps', true, 12.50),
       ('Eté', false, 11.50),
       ('Automne', true, 11.00),
       ('Hiver', true, 12.00);

INSERT INTO choix_pizzas_menu (id_menu, id_pizza)
VALUES
       ((SELECT id FROM menus WHERE nom = 'Printemps'),
              (SELECT id FROM pizzas WHERE nom = 'Margherita')),
       ((SELECT id FROM menus WHERE nom = 'Eté'),
              (SELECT id FROM pizzas WHERE nom = 'Merguez')),
       ((SELECT id FROM menus WHERE nom = 'Automne'),
              (SELECT id FROM pizzas WHERE nom = 'BBQ New York')),
       ((SELECT id FROM menus WHERE nom = 'Hiver'),
              (SELECT id FROM pizzas WHERE nom = '4 fromages')),
       ((SELECT id FROM menus WHERE nom = 'Hiver'),
              (SELECT id FROM pizzas WHERE nom = 'Raclette'));
```

```sql
INSERT INTO choix_desserts_menu (id_menu, id_dessert)
VALUES
       ((SELECT id FROM menus WHERE nom = 'Printemps'),
              (SELECT id FROM desserts WHERE nom = 'Tiramisu')),
       ((SELECT id FROM menus WHERE nom = 'Eté'),
              (SELECT id FROM desserts WHERE nom = 'Glace')),
       ((SELECT id FROM menus WHERE nom = 'Automne'),
              (SELECT id FROM desserts WHERE nom = 'Panna cotta')),
       ((SELECT id FROM menus WHERE nom = 'Hiver'),
              (SELECT id FROM desserts WHERE nom = 'Mousse au chocolat')),
       ((SELECT id FROM menus WHERE nom = 'Hiver'),
              (SELECT id FROM desserts WHERE nom = 'Banane au whisky'));

insert into quartiers (id, nom)
VALUES (1, 'Hotel de Ville'),
(2, 'Bellecours-Ainay-Confluence'),
(3, 'Part-Dieu'),
(4, 'Croix Rousse'),
(5, 'Vieux Lyon'),
(6, 'Foche'),
(7, 'Mermoz'),
(8, 'Etats-Unis');


insert into livreurs (nom, téléphone, quartier, est_en_livraison)
VALUES
('Jean Paul',        '0644889865', 1, false),
('Charles Henry', '0647942304', 2, false),
('Pierre',           '0642989584', 3, false),
('Alice',            '0652495849', 4, false),
('Bob',                    '0640408952', 5, false),
('Rose',                   '0740239944', 6, false),
('Agathe',           '0642342442', 7, false),
('Christophe',       '0642898753', 8, false);


COMMIT;
```

LA FONCTION COMMANDER

```
CREATE OR REPLACE FUNCTION public.commander(
    nom_client character varying,
    liste_menus character varying[],
    liste_desserts_menus character varying[],
    liste_pizzas_menus character varying[],
    liste_pizzas character varying[],
    liste_tailles_pizzas character varying[],
    liste_desserts character varying[],
    liste_boissons character varying[],
    "à_emporter" boolean,
    "téléphone_client" character varying,
    "numéro_rue" character varying,
    rue character varying,
    quartier character varying,
    OUT id_commande integer,
    OUT id_livreur integer)
    RETURNS record
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
DECLARE id_client INTEGER;
DECLARE id_du_menu INTEGER;
DECLARE id_du_quartier INTEGER;

BEGIN
    id_commande := null;
    id_livreur := null;
    -- Enregistrer un client
    if (nom_client is null) then
        raise exception 'Le nom du client est obligatoire';
    end if;
    if not exists (select 1 from clients where nom = nom_client) then
        insert into clients (nom, téléphone) values (nom_client, téléphone_client);
    else
        update clients set téléphone = téléphone_client where nom = nom_client;
    end if;
    id_client := (select id from clients where nom = nom_client);
    -- Enregistrer la commande et verifier la cohérance
    if liste_menus is null and liste_pizzas is null and liste_desserts is null and
liste_boissons is null then
        raise exception 'Il n''y a rien a commander!';
    end if;
    if
        liste_menus is not null and liste_desserts_menus is not null and
cardinality(liste_menus) != cardinality(liste_desserts_menus)
        or
        liste_menus is not null and liste_desserts_menus is null
        or
```

```
        liste_menus is null and liste_desserts_menus is not null
        or
        liste_menus is not null and liste_pizzas_menus is not null and
cardinality(liste_menus) != cardinality(liste_pizzas_menus)
        or
        liste_menus is not null and liste_pizzas_menus is null
        or
        liste_menus is null and liste_pizzas_menus is not null
    then
        raise exception 'Le nombre de menu et de desserts/pizzas par menu doit etre
identique!';
    end if;
    insert into commandes (id_client, montant_total, terminée, payée)
        values (id_client, 0, false, false) returning id into id_commande;
    if liste_menus is not null then
        for i in 1..cardinality(liste_menus) loop
            if not exists (select * from menus where nom = liste_menus[i]) then
                raise exception 'Le menu n''existe pas!';
            end if;
            id_du_menu = (select id from menus where nom = liste_menus[i]);
            insert into menus_par_commande (id_commande, id_menu)
            values (id_commande, id_du_menu);
            if liste_pizzas_menus[i] is null then
                raise exception 'Un menu doit avoir une pizza!';
            end if;
            if not exists (select * from choix_pizzas_menu
                            where id_menu = id_du_menu and  id_pizza = (
                                select id from pizzas where nom =
liste_pizzas_menus[i])) then
                    raise exception 'La pizza n''existe pas dans le menu!';
            end if;
            update menus_par_commande set id_pizza = (
                select id from pizzas where nom = liste_pizzas_menus[i]
            ) where id_menu = id_du_menu;
            if liste_desserts_menus[i] is not null then
                if not exists (select * from choix_desserts_menu
                            where id_menu = id_du_menu and  id_dessert = (
                                select id from desserts where nom =
liste_desserts_menus[i])) then
                    raise exception 'Le déssert n''existe pas dans le menu!';
                end if ;
                update menus_par_commande set id_dessert = (
                    select id from desserts where nom = liste_desserts_menus[i]
                ) where id_menu = id_du_menu;
            end if;
            update commandes
                    set montant_total = montant_total + (SELECT prix FROM menus
WHERE nom = liste_menus[i])
                        where id = id_commande;
        end loop;
    end if;
    -- Extras hors menu
    if liste_pizzas is not null then
```

```
        for i in 1..cardinality(liste_pizzas) loop
            if not exists (select * from pizzas where nom = liste_pizzas[i]) then
                raise exception 'Il y a une pizza dans lise_pizza qui n''existe
pas';
            end if;
            insert into pizzas_par_commande (id_commande, id_pizza, taille) values
(id_commande, (select id from pizzas where nom = liste_pizzas[i]),
liste_tailles_pizzas[i]);
            if (liste_tailles_pizzas[i] = 'petite') then
                update commandes
                    set montant_total = montant_total + (SELECT prix_petite FROM
pizzas WHERE nom = liste_pizzas[i])
                        where id = id_commande;
            elsif (liste_tailles_pizzas[i] = 'moyenne') then
                update commandes
                    set montant_total = montant_total + (SELECT prix_moyenne FROM
pizzas WHERE nom = liste_pizzas[i])
                        where id = id_commande;
            elsif (liste_tailles_pizzas[i] = 'grande') then
                update commandes
                    set montant_total = montant_total + (SELECT prix_grande FROM
pizzas WHERE nom = liste_pizzas[i])
                        where id = id_commande;
            else
                raise exception 'Il manque des taille pour les pizzas hors menu!';
            end if;
        end loop;
    end if;
    if liste_desserts is not null then
        for i in 1..cardinality(liste_desserts) loop
            if not exists (select * from desserts where nom = liste_desserts[i])
then
                raise exception 'Il y a un dessert dans liste_desserts qui
n''existe pas';
            end if;
            update commandes
                    set montant_total = montant_total + (SELECT prix FROM desserts
WHERE nom = liste_desserts[i])
                        where id = id_commande;
            insert into desserts_par_commande (id_commande, id_dessert) values
(id_commande, (select id from desserts where nom = liste_desserts[i]));
        end loop;
    end if;
    if liste_boissons is not null then
        for i in 1..cardinality(liste_boissons) loop
            if not exists (select * from boissons where nom = liste_boissons[i])
then
                raise exception 'Il y a une boisson dans liste_boissons qui
n''existe pas';
            end if;
            update commandes
                    set montant_total = montant_total + (SELECT prix FROM boissons
WHERE nom = liste_boissons[i])
```

```
                        where id = id_commande;
            insert into boissons_par_commande (id_commande, id_boisson) values
(id_commande, (select id from boissons where nom = liste_boissons[i]));
        end loop;
    end if;
    -- Gerer la partie livraison
    if à_emporter then
        if téléphone_client is null or numéro_rue is null or rue is null is null or
quartier is null then
            raise exception 'Les données de livraison sont incomplètes';
        end if;
        id_du_quartier := (select id from quartiers where nom = quartier);
        id_livreur := (select id from livreurs where livreurs.quartier =
id_du_quartier limit 1);
        if id_livreur is null then
            raise exception 'Aucun livreur disponible dans le quartier demandé';
        end if;
        insert into livraisons (id_commande, id_livreur, numéro_rue, rue,
id_quartier, téléphone_client)
        values (id_commande, id_livreur, numéro_rue, rue, id_du_quartier,
téléphone_client);
   end if;
 END;
$BODY$;
```

AUTRES REQUETES

```
-- passer une commande
SELECT * from public.commander(
    nom_client => 'John Doe',
    -- LES MENUS ET LES OPTIONS DES MENUS
    liste_menus => array['Hiver', 'Eté'],
    liste_desserts_menus => array['Banane au whisky', 'Glace'],
    liste_pizzas_menus => array ['4 fromages', 'Merguez'],
    -- LES EXTRAS
    liste_pizzas => array['Raclette'],
    liste_tailles_pizzas => array['petite'],
    liste_desserts => array['Tiramisu'],
    liste_boissons => array['1664', '1664', '1664', 'Sprite'],
    -- DETAILS COMMANDE
    à_emporter => true,
    téléphone_client => '07 88 09 12 35',
    numéro_rue => '35B',
    rue => 'Rue Paul Santy',
    quartier => 'Mermoz'
)
```

	id_commande integer	id_livreur integer
1	4	7

```
-- liste des menus commandés
select menus.nom as nom_menu, pizzas.nom as nom_pizza,
desserts.nom as nom_dessert, fromage, menus.prix
    from menus_par_commande, menus, pizzas, desserts
where
    id_commande = 4
    and id_menu = menus.id
    and pizzas.id = id_pizza
    and desserts.id = id_dessert
```

	nom_menu character varying (255)	nom_pizza character varying (255)	nom_dessert character varying (255)	fromage boolean	prix numeric (5,2)
1	Eté	Merguez	Glace	false	11.50
2	Hiver	4 fromages	Banane au whisky	true	12.00

```
-- liste pizzas en extra
select pizzas.nom, taille from pizzas_par_commande, pizzas
where
    pizzas.id = id_pizza
```

	nom character varying (255)	taille character varying (255)
1	Raclette	petite

```
and id_commande = 4
```

```sql
-- liste desserts en extra
select desserts.nom from desserts_par_commande, desserts
where
    desserts.id = id_dessert
    and id_commande = 4
```

	nom character varying (255)
1	Tiramisu

```sql
-- liste boissons en extra
select boissons.nom from boissons_par_commande, boissons
where
    boissons.id = id_boisson
    and id_commande = 4
```

	nom character varying (255)
1	1664
2	1664
3	1664
4	Sprite

```sql
-- le total de la commande
select montant_total from commandes where id = 4
```

	montant_total numeric (10,2)
1	45.00

```sql
-- Un livreur est arrivée chez un client
BEGIN TRANSACTION;
    update public.commande set terminée = true, payée = true where id = 8;
    update public.livraison set date_heure_livraison = now() where id_commande = 8;
COMMIT;
```

```sql
-- Example de requête pour le retour d'un livreur à la pizzeria après une livraison
BEGIN TRANSACTION;
    update public.livreurs set est_en_livraison = false
    where nom = 'Rose'
COMMIT;
```

```sql
-- liste des boisons par nb vente decroissant
select boissons.nom, count(boissons_par_commande.id) as nb
from public.boissons, boissons_par_commande
where boissons.id = id_boisson
GROUP BY boissons.nom
ORDER BY nb desc
```

	nom character varying (255)	nb bigint
1	1664	12
2	Sprite	4

NOTES

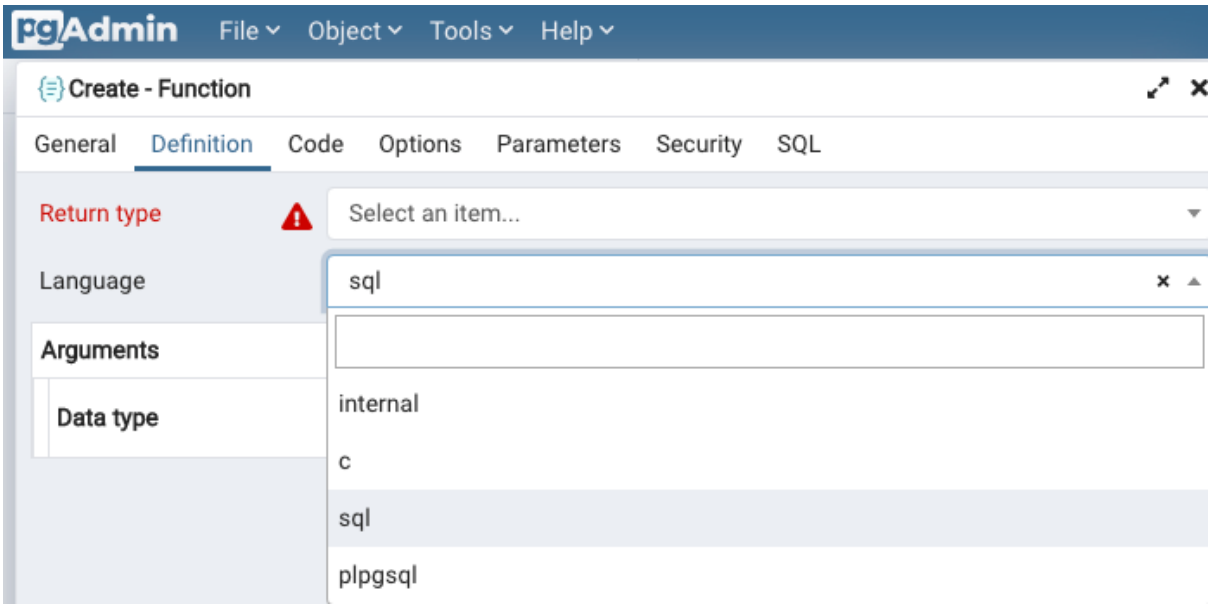
Nous avons choisi comme dernière requête les boissons les plus vendus car une requête pour les pizzas les plus vendus, en sachant que les pizzas peuvent être en menu ou en extra aurait été un peu plus compliqué sans plpgsql ou autre.

Nous avons apprécié le langage plpgsql pour notre fonction « commander » uniquement. PostgreSQL est en effet très puissant mais les messages d'erreurs de syntaxe de plpgsql n'ont pas été facile à comprendre pour nous débutants.

Aussi nous avons choisis une fonction et non pas une procédure stockée car nous voulions pouvoir immédiatement utiliser le résultat de notre fonction commander :

```
select montant_total from commandes where id = (SELECT id_commande from
public.commander(
    nom_client => ...
))
```

Mais seules les procédures stockées peuvent COMMIT leur transaction avant de retourner ; ce n'est pas possible avec une fonction sans utiliser des techniques avancées que nous n'avons pas encore trouvé. Nous avons remarqué qu'il est possible d'écrire des fonctions pour PostgreSQL en langage C directement ; peut-être que cela pourrait nous aider à contourner ce genre de problème.



Mais … cela dépasserait le cadre ce TP.