Εθνικό Μετσόβιο Πολυτεχνείο

ΣΕΜΦΕ-8ο Εξάμηνο

Βάσεις Δεδομένων

Γεώργιος Ανδρονίκου(ge20720)
georgios.andronikou@gmail.com

Θεόδωρος Μπέκος(ge20034)
theompe2018@gmail.com

GitHub Repository Masterchef Database

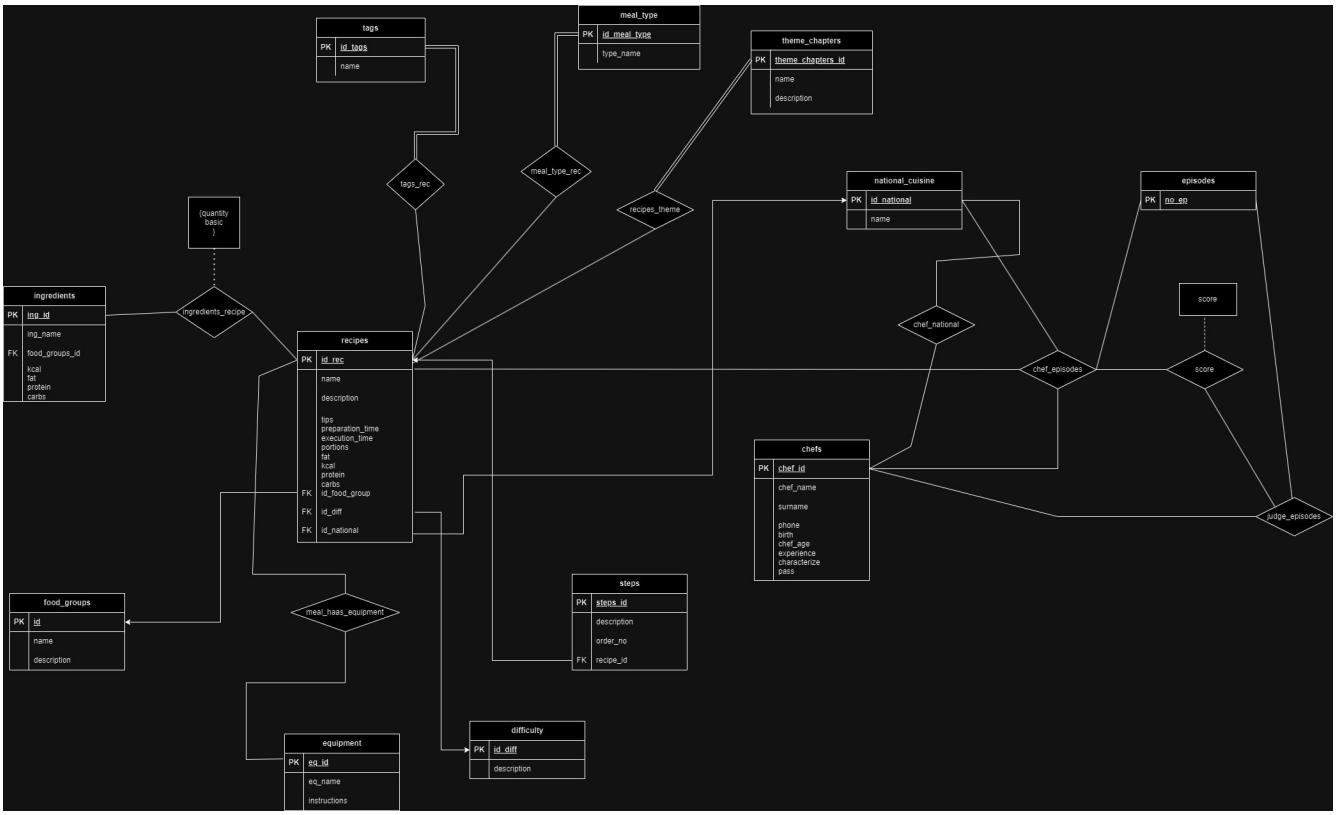
Περιεχόμενα

Σελίδα

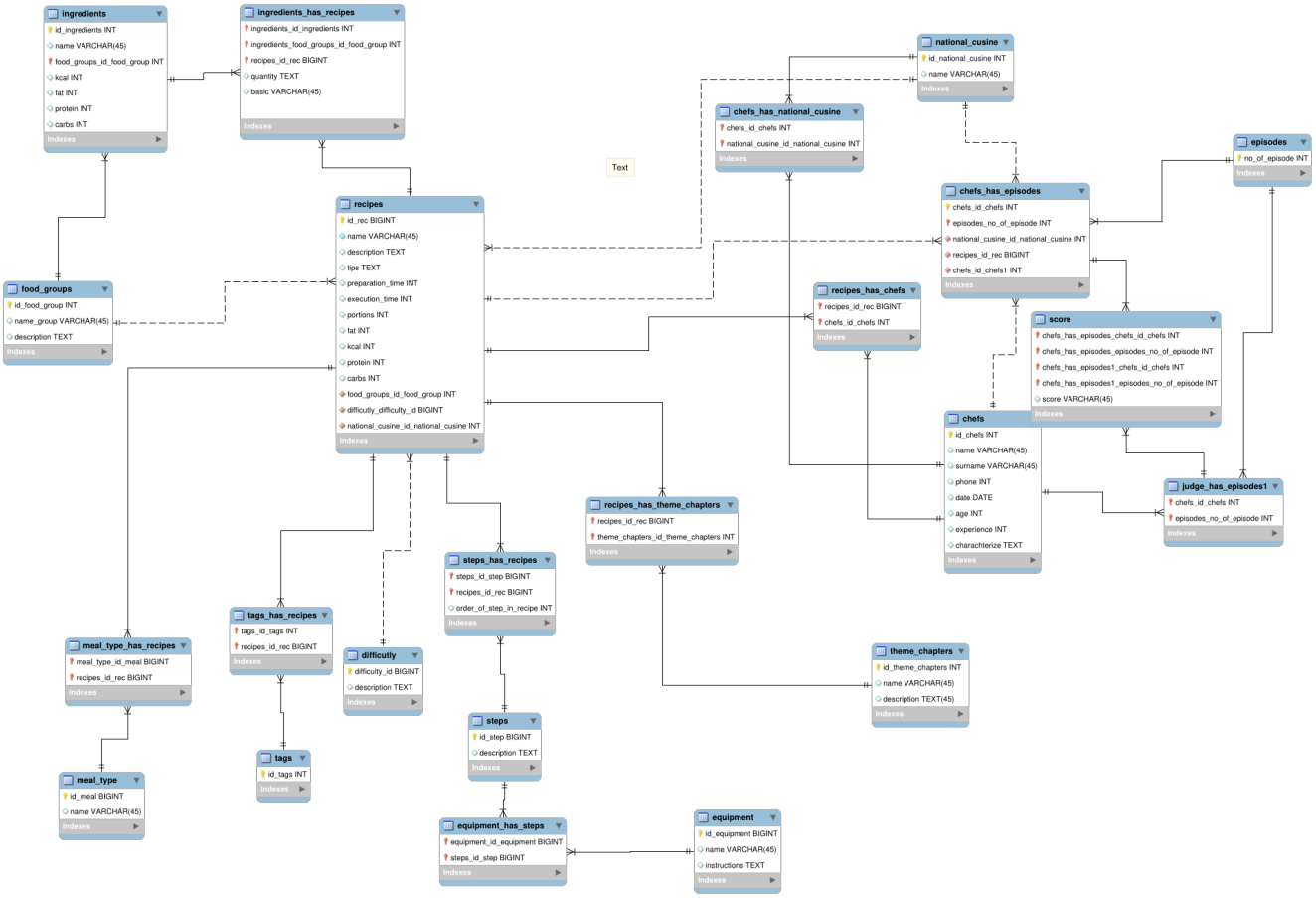
1 Εισαγωγικά

Στην παρούσα αναφορά, θα περιγράψουμε τη διαδικασία που ακολουθήσαμε για την υλοποίηση και διαχείριση της βάσης δεδομένων μας.Χρησιμοποιήσαμε ως σύστημα διαχείρησης δεδομένων την PostgreSQL και ως εργαλείο διαχείρησης βάσεων δεδομένων με γραφικό περιβάλλον χρήστη , χρησιμοποιήσαμε το dbeaver.
Σημείωση:Για οποιοδήποτε script χρειάζεται να τρέξει πρέπει να αλλαχτούν τα απαραίτητα directories αλλά connection με την βάση.

2 ER diagram



3 RelationalD Diagram



4 Περιορισμοί

4.1 Primary Keys for Tables(Πρωτεύον κλειδιά για πίνακες)

- **food_groups**: `id`
- **difficulty**:`id_diff`
- **national_cuisine**: `id_national`
- **recipes**: `id_rec` .
- **meal_type**: `id_meal_type`
- **tags**: `id_tags`
- **ingredients**: `ing_id`
- **equipment**: `eq_id`
- **steps**: `steps_id`
- **theme_chapters**: `theme_chapters_id`
- **episodes**: `no_ep`
- **chefs**: `chef_id`

Εκτός από τους πίνακες, οι σχέσεις που περιλαμβάνονται στον παρεχόμενο κώδικα SQL δημιουργούν επίσης πρωτεύοντα κλειδιά. Αυτά τα πρωτεύοντα κλειδιά είναι συνδυασμοί από ξένα κλειδιά που εξασφαλίζουν τη μοναδικότητα κάθε καταχώρισης στις σχέσεις. Ακολουθούν οι σχέσεις και τα αντίστοιχα πρωτεύοντα κλειδιά τους.

4.2 Σχέσεις με Πρωτεύοντα Κλειδιά

- **meal_type_rec**:
  - PK: (`id_rec`, `id_type`)
- **tags_rec**:
  - PK: (`id_tags`, `id_rec`)
- **ingredient_recipes**:
  - PK: (`id_ing`, `id_rec`)
- **recipes_has_equipment**:
  - PK: (`recipes_id`, `equipment_id`)
- **recipes_theme**:
  - PK:(`id_rec`, `id_theme`)

- **chefs_episodes**:
  - PK: (`chef_id`, `chef_no_ep`)
- **judge_episodes**:
  - PK: (`judge_id`, `judge_no_ep`)
- **score**:
  - PK: (`chef_id`, `chef_no_ep`, `judge_id`, `judge_no_ep`)
- **chef_national**:
  - PK: (`id_chef`, `id_national`)

Στον παρεχόμενο κώδικα SQL, δημιουργούνται διάφορα ξένα κλειδιά (foreign keys) που εξασφαλίζουν την αναφορική ακεραιότητα μεταξύ των πινάκων. Παρακάτω παρατίθεται η επεξήγηση όλων των ξένων κλειδιών στον κώδικα.

4.3 Πίνακες με Ξένα Κλειδιά

- **recipes**:
  - `id_food_group` αναφέρεται στον πίνακα `food_groups`.
  - `id_diff` αναφέρεται στον πίνακα `difficulty`.
  - `id_national` αναφέρεται στον πίνακα `national_cuisine`.
- **meal_type_rec**:
  - `id_rec` αναφέρεται στον πίνακα `recipes`.
  - `id_type` αναφέρεται στον πίνακα `meal_type`.
- **tags_rec**:
  - `id_tags` αναφέρεται στον πίνακα `tags`.
  - `id_rec` αναφέρεται στον πίνακα `recipes`.
- **ingredients**:
  - `food_groups_id` αναφέρεται στον πίνακα `food_groups`.
- **ingredient_recipes**:
  - `id_ing` αναφέρεται στον πίνακα `ingredients`.
  - `id_rec` αναφέρεται στον πίνακα `recipes`.
- **recipes_has_equipment**:
  - `recipes_id` αναφέρεται στον πίνακα `recipes`.
  - `equipment_id` αναφέρεται στον πίνακα `equipment`.

- **steps**:
  - `recipe_id` αναφέρεται στον πίνακα `recipes`.
- **recipes_theme**:
  - `id_rec` αναφέρεται στον πίνακα `recipes`.
  - `id_theme` αναφέρεται στον πίνακα `theme_chapters`.
- **chefs_episodes**:
  - `chef_id` αναφέρεται στον πίνακα `chefs`.
  - `chef_no_ep` αναφέρεται στον πίνακα `episodes`.
  - `id_national` αναφέρεται στον πίνακα `national_cuisine`.
  - `rec_id` αναφέρεται στον πίνακα `recipes`.
- **judge_episodes**:
  - `judge_id` αναφέρεται στον πίνακα `chefs`.
  - `judge_no_ep` αναφέρεται στον πίνακα `episodes`.
- **score**:
  - `chef_id`, `chef_no_ep` αναφέρονται στον πίνακα `chefs_episodes`.
  - `judge_id`, `judge_no_ep` αναφέρονται στον πίνακα `judge_episodes`.
- **chef_national**:
  - `id_chef` αναφέρεται στον πίνακα `chefs`.
  - `id_national` αναφέρεται στον πίνακα `national_cuisine`.

4.4 Περιορισμοί ανά Πίνακα

Στον παρεχόμενο κώδικα SQL, χρησιμοποιούνται διάφοροι περιορισμοί (constraints) όπως UNIQUE, NOT NULL, CHECK, κ.λπ., για να εξασφαλιστεί η ακεραιότητα των δεδομένων. Παρακάτω παρατίθεται η επεξήγηση όλων των περιορισμών στον κώδικα.
Για όλες τις στήλες έχουμε να είναι not null , ενώ για τα primary key οι περιορισμοί not null και unique εφαρμόζονται απο το PK.Επιπλέον περιορισμούς έχουμε στα υλικά και στις συνταγές , όπου για τις διατροφικές αξίες έχουμε να είναι ≥ 0.Επίσης ο χρόνος εκτέλεσης και προετοιμασίας πάλι έχουν περιορισμό να είναι > 0.Στον πίνακα Chef ελέγχουμε η ηλικία του chef να βγάζει νόημα με την ημερομηνία γέννησης πάλι με την χρήση περιορισμού.Επίσης για τα κινητά έχουμε περιορισμο να έχουν μήκος 10 αριθμούς

4.5 Views(Προβολές)

Σε αυτή την ενότητα περιγράφονται οι προβολές (views) που δημιουργήθηκαν στη βάση δεδομένων για την παρουσίαση συγκεκριμένων πληροφοριών. Οι προβολές επιτρέπουν τη σύνθεση και την οργάνωση των δεδομένων με τρόπο που να διευκολύνει την ανάλυση και την παρουσίαση.

Συνταγές και τα Συστατικά τους

Η προβολή `recipe_ingredients` δημιουργήθηκε για να παρουσιάσει κάθε συνταγή μαζί με τα συστατικά της σε μία λίστα. Η συγκεκριμένη προβολή συνδυάζει τις πληροφορίες από τους πίνακες `recipes`, `ingredient_recipes` και `ingredients`.

Listing 1: Προβολή Συνταγών και των Συστατικών τους

```
CREATE VIEW recipe_ingredients AS
SELECT
    r.name AS recipe_name,
    array_agg(i.ing_name) AS ingredients
FROM
    recipes r
JOIN
    ingredient_recipes ir ON r.id_rec = ir.id_rec
JOIN
    ingredients i ON ir.id_ing = i.ing_id
GROUP BY
    r.name;
```

Η προβολή αυτή επιστρέφει τα ονόματα των συνταγών και μια λίστα με τα ονόματα των συστατικών τους. Η χρήση της συνάρτησης `array_agg` επιτρέπει τη συγκέντρωση των συστατικών σε μία λίστα για κάθε συνταγή.

Συνολικοι Πόντοι ανά Σεζόν ανά Σεφ

Η προβολή `chef_total_points_per_season` δημιουργήθηκε για να παρουσιάσει τους συνολικούς πόντους κάθε σεφ ανά σεζόν. Η προβολή αυτή χρησιμοποιεί έναν κοινό πίνακα εκφράσεων (CTE) για να καθορίσει τα επεισόδια που ανήκουν σε κάθε σεζόν και στη συνέχεια υπολογίζει τους συνολικούς πόντους για κάθε σεφ ανά σεζόν.

Listing 2: Προβολή Συνολικών Πόντων ανά Σεζόν ανά Σεφ

```
CREATE VIEW chef_total_points_per_season AS
WITH season_episodes AS (
    SELECT 1 AS season, 1 AS start_ep, 10 AS end_ep UNION ALL
    SELECT 2 AS season, 11 AS start_ep, 20 AS end_ep UNION ALL
    SELECT 3 AS season, 21 AS start_ep, 30 AS end_ep UNION ALL
    SELECT 4 AS season, 31 AS start_ep, 40 AS end_ep UNION ALL
    SELECT 5 AS season, 41 AS start_ep, 50 AS end_ep
),
season_scores AS (
    SELECT
        s.chef_id,
        se.season,
        SUM(s.score) AS total_score
    FROM
        score s
    JOIN
        chefs_episodes ce ON s.chef_id = ce.chef_id AND s.chef_no_ep = ce.chef_no_ep
    JOIN
        season_episodes se ON s.chef_no_ep BETWEEN se.start_ep AND se.end_ep
    GROUP BY
```

```
        s.chef_id, se.season
)
SELECT
    c.chef_id,
    c.chef_name,
    c.surname,
    ss.season,
    ss.total_score
FROM
    chefs c
JOIN
    season_scores ss ON c.chef_id = ss.chef_id
ORDER BY season, total_score;
```

Αυτή η προβολή χρησιμοποιεί δύο κοινές εκφράσεις:

- **season_episodes**: Καθορίζει τα επεισόδια που ανήκουν σε κάθε σεζόν.
- **season_scores**: Υπολογίζει τους συνολικούς πόντους για κάθε σεφ ανά σεζόν.

Στη συνέχεια, η τελική επιλογή (SELECT) συνδυάζει τα υπολογισμένα αποτελέσματα με τις πληροφορίες των σεφ από τον πίνακα `chefs` και επιστρέφει τους συνολικούς πόντους και τα στοιχεία των σεφ ανά σεζόν, ταξινομημένα κατά σεζόν και συνολικούς πόντους.
Αυτές οι προβολές διευκολύνουν την ανάλυση των δεδομένων και παρέχουν συγκεντρωτικές πληροφορίες που είναι χρήσιμες για την παρουσίαση και την κατανόηση των δεδομένων της βάσης.

4.6 Triggers

Σε αυτή την ενότητα, θα αναλύσουμε δύο triggers που έχουν δημιουργηθεί στη βάση δεδομένων. Ο πρώτος trigger αφορά τον υπολογισμό των διατροφικών στοιχείων μιας συνταγής και ο δεύτερος την δημιουργία ρόλου χρήστη σεφ κατά την εισαγωγή ενός νέου σεφ.
Trigger Υπολογισμού Διατροφικών Στοιχείων Συνταγής
Ο trigger `update_recipe_nutrition` εκτελείται μετά από κάθε εισαγωγή ή ενημέρωση στη σχέση `ingredient_recipes`. Η λειτουργία του είναι να υπολογίζει τα συνολικά διατροφικά στοιχεία (θερμίδες, λίπη, πρωτεΐνες, υδατάνθρακες) μιας συνταγής βάσει των συστατικών της.

Listing 3: Trigger Υπολογισμού Διατροφικών Στοιχείων Συνταγής

```
CREATE OR REPLACE FUNCTION convert_quantity_to_factor(quantity_text TEXT) RETURNS FLOAT AS
DECLARE
    numeric_part FLOAT;
    unit_part TEXT;
BEGIN
    -- Extract numeric part from the quantity text
    numeric_part := substring(quantity_text from '^[0-9]+')::FLOAT;

    -- Determine the unit (assuming only 'g' or 'ml' are used)
    unit_part := substring(quantity_text from '[a-zA-Z]+$');

    -- Convert based on unit
    IF unit_part = 'g' THEN
        RETURN numeric_part / 100.0;
    ELSIF unit_part = 'ml' THEN
        RETURN numeric_part / 100.0;
    ELSE
```

```
        -- Default case to handle unexpected unit
        RETURN NULL;
    END IF;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION recipes_nutrition_insert_trigger() RETURNS TRIGGER AS $$
DECLARE
    total_kcal FLOAT := 0;
    total_fat FLOAT := 0;
    total_protein FLOAT := 0;
    total_carbs FLOAT := 0;
    quantity_factor FLOAT;
    rec RECORD;
BEGIN
    -- Calculate total nutritional values
    FOR rec IN (
        SELECT i.ing_id, i.kcal, i.fat, i.protein, i.carbs, ir.quantity
        FROM ingredients i
        JOIN ingredient_recipes ir ON i.ing_id = ir.id_ing
        WHERE ir.id_rec = NEW.id_rec
    ) LOOP
        quantity_factor := convert_quantity_to_factor(rec.quantity);

        -- Debugging output
        RAISE NOTICE 'Ingredient: %, Quantity: %, Factor: %', rec.ing_id, rec.quanti

        -- Ensure the quantity factor is not null
        IF quantity_factor IS NOT NULL THEN
            total_kcal := total_kcal + (rec.kcal * quantity_factor);
            total_fat := total_fat + (rec.fat * quantity_factor);
            total_protein := total_protein + (rec.protein * quantity_factor);
            total_carbs := total_carbs + (rec.carbs * quantity_factor);
        ELSE
            RAISE NOTICE 'Skipping ingredient % due to null quantity factor', rec.ir
        END IF;
    END LOOP;

    -- Update the recipe row with calculated totals
    UPDATE recipes
    SET kcal = total_kcal,
        fat = total_fat,
        protein = total_protein,
        carbs = total_carbs
    WHERE id_rec = NEW.id_rec;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Drop existing trigger if it exists
DROP TRIGGER IF EXISTS update_recipe_nutrition ON ingredient_recipes;

-- Create the trigger to call the function after insert or update
CREATE TRIGGER update_recipe_nutrition
AFTER INSERT OR UPDATE ON ingredient_recipes
```

```
FOR EACH ROW
EXECUTE FUNCTION recipes_nutrition_insert_trigger();
```

Η συνάρτηση convert_quantity_to_factor μετατρέπει την ποσότητα των συστατικών από κείμενο σε αριθμητική τιμή και τη διαιρεί με το 100 για να προσαρμόσει την ποσότητα. Η συνάρτηση recipes_nutrition_insert_trigger χρησιμοποιεί αυτή τη μετατροπή για να υπολογίσει τα συνολικά διατροφικά στοιχεία της συνταγής και ενημερώνει τη σχέση recipes με τα αποτελέσματα.

Trigger Δημιουργίας Χρήστη Σεφ

Ο trigger after_chef_insert εκτελείται μετά από κάθε εισαγωγή στη σχέση chefs. Η λειτουργία του είναι να δημιουργεί έναν νέο ρόλο χρήστη με δικαιώματα εισαγωγής και ενημέρωσης στις συνταγές και στους σεφ.

Listing 4: Trigger Δημιουργίας Χρήστη Σεφ

```
CREATE OR REPLACE FUNCTION create_chef_user()
RETURNS TRIGGER AS $$
BEGIN

    -- Create a new role with the chef_id as both the username and password

    EXECUTE format('CREATE ROLE %I WITH LOGIN PASSWORD %L', NEW.chef_id, NEW.pas);

    -- Grant INSERT and UPDATE privileges on the recipes table to the new role

    EXECUTE format('GRANT INSERT, UPDATE ON TABLE recipes TO %I', NEW.chef_id);

    -- Grant INSERT and UPDATE privileges on the chefs table to the new role

    EXECUTE format('GRANT INSERT, UPDATE ON TABLE chefs TO %I', NEW.chef_id);

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Create the trigger to call the function after insert
CREATE TRIGGER after_chef_insert
AFTER INSERT ON chefs
FOR EACH ROW
EXECUTE FUNCTION create_chef_user();
```

Η συνάρτηση create_chef_user δημιουργεί ένα νέο ρόλο χρήστη με το chef_id ως όνομα χρήστη και κωδικό πρόσβασης, και χορηγεί δικαιώματα εισαγωγής και ενημέρωσης στις σχέσεις recipes και chefs.

5 Indexes

Τα indexes στη βάση δεδομένων χρησιμοποιούνται για να βελτιώσουν την απόδοση των ερωτημάτων επιτρέποντας ταχύτερη πρόσβαση στα δεδομένα. Παρακάτω περιγράφουμε τα indexes που χρησιμοποιούνται στη βάση δεδομένων μας και τους λόγους για τους οποίους τα χρησιμοποιούμε.

5.1 `tags_rec`

- `id_rec,id_tags`: Δημιουργήσαμε αυτά τα indexes για να βελτιώσουμε την απόδοση των ερωτημάτων που αναζητούν εγγραφές στον πίνακα `tags_rec` με βάση το `id_rec` `id_tags`. Αυτό είναι ιδιαίτερα χρήσιμο όταν θέλουμε να βρούμε όλες τις ετικέτες που σχετίζονται με μια συγκεκριμένη συνταγή και το αντίθετο.

5.2 `chefs_episodes`

- `no_ep`: Δημιουργήσαμε αυτό το index για να επιταχύνουμε την αναζήτηση των εγγραφών στον πίνακα `chefs_episodes` με βάση το `chef_no_ep`. Αυτό είναι σημαντικό για την ταχύτερη εύρεση των επεισοδίων στα οποία συμμετείχε ένας σεφ και για το ερώτημα 3.6

5.3 `recipes_has_equipment`

- `idx_recipes_id`: Δημιουργήσαμε αυτό το index για να επιταχύνουμε την αναζήτηση των συνταγών με βάση το id στο πίνακα σχέσης με τον εξοπλισμό. Είναι χρήσιμο για το ερώτημα 3.8

5.4 ingredients

- `idx_ingredients_name`: Δημιουργήσαμε αυτό το index για να επιταχύνουμε την αναζήτηση των υλικών με βάση το όνομα. Αυτό είναι ιδιαίτερα χρήσιμο όταν θέλουμε να βρούμε ένα συγκεκριμένο υλικό στη βάση δεδομένων.

5.5 score

- `idx_score_chef_id`, `idx_score_chef_no_ep`, `idx_score_judge_id`, `idx_score_judge_no_ep`: Αυτά τα indexes δημιουργήθηκαν για να βελτιώσουν την απόδοση των ερωτημάτων στον πίνακα `score`. Δεδομένου ότι ο πίνακας αυτός περιέχει 1500 εγγραφές, τα indexes αυτά επιτρέπουν την ταχύτερη αναζήτηση των βαθμολογιών με βάση τους σεφ και τους κριτές.

Με τη δημιουργία αυτών των indexes, επιτυγχάνουμε σημαντική βελτίωση στην απόδοση των ερωτημάτων στη βάση δεδομένων μας, καθιστώντας τις αναζητήσεις και τις ανακτήσεις δεδομένων πιο αποδοτικές και γρήγορες.

6 Δημιουργία βάσης

Αρχικά ενωνώμαστε στην SQL με τα ακόλουθα στοιχεία:

- **Server:**localhost
- **Database:**postgres
- **Port:**5432
- **Username:**postgres
- **Password:**postgres(default) or no password or the password of the device

ή διαφορετικά σε Unix με την εντολή "sudo -u postgres psql" στην terminal.

Υπάρχει η περίπτωση να έχουμε πρόβλημα με την σύνδεση στη postgres.
Αυτό ήταν ένα πρόβλημα που αντιμετωπίσαμε.Για την επίλυση του αρχικά εντοπίζουμε το file "pg_hba.conf" με την βοήθεια της εντολής:

```
sudo -u postgres psql -c "SHOW hba_file;"
```

Από το τερματικό , ή μόνο με την εντολή που βρίσκεται σε εισαγωγικά από interactive terminal της postgres (psql).
Στην συνέχεια μπορούμε να ανοίξουμε το αρχείο(nano-ή σε οποιονδήποτε άλλο editor) με τις ιδιότητες του superuser(sudo) διότι δεν έχει write permissions.
Αφού ανοίξουμε το αρχείο θα δούμε την παρακάτω μορφή:

```
# "local" is for Unix domain socket connections only
local   all             all                                     ident
# IPv4 local connections:
host    all             all             127.0.0.1/32            ident
# IPv6 local connections:
host    all             all             ::1/128                 ident
```

Εμείς πρέπει να αλλάξουμε τις ρυθμίσεις ident σε md5 όπως φαίνεται παρακάτω:

```
# "local" is for Unix domain socket connections only
local   all             all                                     md5
# IPv4 local connections:
host    all             all             127.0.0.1/32            md5
# IPv6 local connections:
host    all             all             ::1/128                 md5
```

Επίσης ένα άλλο πρόβλημα που συνατνήσαμε ήταν το dbeaver να κάνει crash όταν προσθέσαμε τις εικόνες στα υλικά διότι δεν μπορούσε να κάνει fetch τα δεδομένα για να τα δούμε μέσω του interface,
Για αυτό το πρόβλημα πρώτα τρεχουμε την εντολή:

```
sudo find / -name "dbeaver.ini" 2>/dev/null
```

για να βρούμε το directory του αρχείου dbeaver.ini
Στην συνέχει πάλι ανοίγουμε το αρχείο με την εντολή nano και superuser privilages με την εντολή:

```
sudo nano /path/to/dbeaver.ini
```

Και αλλάζουμε τις ρυθμίσεις:

```
-Xms64m
-Xmx1024m
```

Σε

```
-Xms1024m
-Xmx4096m
```

Στη συνέχεια δημιουργούμε την βάση από την interactive terminal psql και συνδεόμαστε σε αυτή.

```
postgres=# CREATE DATABASE masterchef;
postgres=# \c masterchef
```

ή αλλιώς με την βοήθεια του dbeaver δημιουργούμε ένα καινούριο connection και συμπληρώνουμε τα στοιχεία ποιο πάνω με τη μόνη διαφορά database:masterchef

Στη συνέχεια τρέχουμε το Sql Script "create_tables_triggres.sql" για την δημιουργία των tables, constrains και triggers.
Το script μπορούμε να το τρέξουμε είτε από psql είτε από dbeaver αφού συνδεθούμε στην βάση που δημιουργήσαμε.
Στο τέλος του της αναφοράς βρίσκεται το DDL script

7 Popuate Tables

Για την εισαγωγή των δεδομένων , θα χρησιμοποιήσουμε ένα python script, για να κάνουμε import τα δεδομένα στη βάση.Εναλλακτικά μπορούμε να χρησιμοποιήσουμε την εντολή "copy" της postgres ή από το γραφικό περιβάλλον του dbever που χρησιμοποιήσαμε και εμείς αρχικά για να επιβεβαιώσουμε ότι η δομή των δεδομένων ήταν ορθή.
Το script που θα τρέχουμε αρχικά είναι το **populate.py**
Τα δεδομένα ήδη βρίσκονται σε αρχεία csv και εμείς τώρα με το script θα τα ανεβάσουμε στη βάση με την σωστή σειρά, ουτως ώστε να μην υπάρχει πρόβλημα με τους περιορισμούς των ξένων κλειδιών.Για παράδειγμα δεν μπορούμε να ανεβάσουμε τα δεδομένα στον πίνακα recipes προτού ανεβάσουμε τα δεδομένα στους πίνακες food_groups, difficulty και national_cusine καθώς και τα 3 έχουν foreign key στο recipes.
Για το script populate.py θα χρειαστούμε τις βιβλιοθήκες :

- **psycop2:**Δημιουργεί μια σύνδεση με την βάση δεδομένων(psycop2.connect) χρησιμοποιώντας τις παραμέτρους της βάσης(φαίνονται στην αρχή του αρχείου).Η εντολή "conn.cursor(),cursor.execute()" μας δίνουν την δυνατότητα να τρέχουμε εντολές sql.Τέλος οι "conn.commit(),conn.rollback()" μας βοηθούν να κάνουμε τις απαραίτητες συναλλαγές με την βάση δεδομένων, και η execute.values() από την (psycopg2.extras) μας δίνει την δυνατότητα για μαζική εισαγωγή δεδομένων

- **pandas:**Η συγκερκιμένη βιβλιοθήκη μας βοηθάει να κάνουμε ανάγνωση των csv αρχείων (pd.read_csv()) σε ένα dataframe.Επίσης, η εντολή "data.to_numpy().tolist" μας βοηθάει να δημιουργήσουμε μια λίστα από tuples και να τα ανεβάσουμε ένα ένα στην βάση.

- **os:** Χρειαζόμαστε αυτή την βιβλιοθήκη για να έχουμε πρόσβαση στο σύστημα του υπολογιστή και να πάρουμε τα απαραίτητα directories στα οποία βρίσκονται τα δεδομένα.

Μπορούμε να κατεβάσεουμε τις συγκεκριμένες βιβλιοθήκες με την βοήθεια του πακέτου pip της python.Αρχικά εκτελούμε

```
sudo dnf install pip
```

ή "sudo apt-get intall pip" για να κατεβάσουμε το πακέτο pip.Στην συνέχει κατεβάζουμε τις απαραίτητες βιβλιοθήκες με την εντολή:

```
pip install "library"
```

```
import os
import pandas as pd
import psycopg2
from psycopg2.extras import execute_values

# Database connection parameters
DB_HOST = 'localhost'
DB_NAME = 'masterchef'
DB_USER = 'postgres'
DB_PASSWORD = 'postgres'

# Base directory for CSV files
BASE_DIR = '/home/georgeandro/george_home/semfe/8o/database/ergasia/git_repo_masterchef'

# Define the CSV file paths
CSV_PATHS = {
    'food_groups': os.path.join(BASE_DIR, 'foodgroups.csv'),
    'difficulty': os.path.join(BASE_DIR, 'difficulty.csv'),
    'national_cuisine': os.path.join(BASE_DIR, 'national_cusine.csv'),
    'recipes': os.path.join(BASE_DIR, 'Recipes_final.csv'),
    'meal_type': os.path.join(BASE_DIR, 'meal_type.csv'),
    'meal_type_rec': os.path.join(BASE_DIR, 'Recipe_Meal_Type_Relationships.csv'),
    'tags': os.path.join(BASE_DIR, 'Tags.csv'),
    'tags_rec': os.path.join(BASE_DIR, 'Recipe_Tags_Relationships.csv'),
    'ingredients': os.path.join(BASE_DIR, 'Ingredients_final.csv'),
    'ingredient_recipes': os.path.join(BASE_DIR, 'Relation_Ing_Rec.csv'),
    'equipment': os.path.join(BASE_DIR, 'equipment.csv'),
    'recipes_has_equipment': os.path.join(BASE_DIR, 'equipment_recipe_relation.csv'),
    'steps': os.path.join(BASE_DIR, 'steps.csv'),
    'theme_chapters': os.path.join(BASE_DIR, 'theme_chapters.csv'),
    'recipes_theme': os.path.join(BASE_DIR, 'recipes_theme.csv'),
    'episodes': os.path.join(BASE_DIR, 'episodes.csv'),
    'chefs': os.path.join(BASE_DIR, 'chef.csv'),
    'chef_national': os.path.join(BASE_DIR, 'chefs_has_national.csv')
}

# Function to populate a table from a CSV file
def populate_table(cursor, table_name, csv_file):
    data = pd.read_csv(csv_file)
    columns = ', '.join(data.columns)
    # Convert data to a list of tuples with native Python types
    values = data.to_numpy().tolist()
    insert_query = f'INSERT INTO {table_name} ({columns}) VALUES %s'
    execute_values(cursor, insert_query, values)

def main():
    try:
        # Connect to the database
```

```
        conn = psycopg2.connect(
            host=DB_HOST,
            dbname=DB_NAME,
            user=DB_USER,
            password=DB_PASSWORD
        )
        cursor = conn.cursor()

        # Import data in the specified order
        tables_in_order = [
            ('food_groups', CSV_PATHS['food_groups']),
            ('difficulty', CSV_PATHS['difficulty']),
            ('national_cuisine', CSV_PATHS['national_cuisine']),
            ('recipes', CSV_PATHS['recipes']),
            ('meal_type', CSV_PATHS['meal_type']),
            ('meal_type_rec', CSV_PATHS['meal_type_rec']),
            ('tags', CSV_PATHS['tags']),
            ('tags_rec', CSV_PATHS['tags_rec']),
            ('ingredients', CSV_PATHS['ingredients']),
            ('ingredient_recipes', CSV_PATHS['ingredient_recipes']),
            ('equipment', CSV_PATHS['equipment']),
            ('recipes_has_equipment', CSV_PATHS['recipes_has_equipment']),
            ('steps', CSV_PATHS['steps']),
            ('theme_chapters', CSV_PATHS['theme_chapters']),
            ('recipes_theme', CSV_PATHS['recipes_theme']),
            ('episodes', CSV_PATHS['episodes']),
            ('chefs', CSV_PATHS['chefs']),
            ('chef_national', CSV_PATHS['chef_national'])
        ]

        for table, csv_file in tables_in_order:
            print(f'Populating {table} from {csv_file}...')
            populate_table(cursor, table, csv_file)

        # Commit the transaction
        conn.commit()

    except Exception as e:
        print(f"An error occurred: {e}")
        if conn:
            conn.rollback()

    finally:
        if cursor:
            cursor.close()
        if conn:
            conn.close()

if __name__ == '__main__':
    main()
```

Στη συνέχεια θα τρέξουμε ακόμη ένα script , το ranfom_script2.py για να κάνουμε την κλήρωση για τον διαγωνισμό του masterchef.Έχουμε 50 επεισόδια (10 επεισόδια κάθε σεζον) , 53 συνταγές και 50 chef.Σε κάθε επεισόδιο , επιλέγουμε 10 εθνικές κουζίνες και 1 μάγειρα αντιπρόσωπο για κάθε εθνική κουζίνα και μια συνταγή για την εθνική κουζίνα αυτή.Επίσης επιλέγουμε και 3 κριτές από τους εναπομείνονα μάγειρες.

- Και σε αυτό το script , χρησιμοποιούμε την βιβλιοθήκη psycopg2 για τις ίδιες λειτουργείες όπως και πιο πάνω.Δηλαδή για να συνδεθούμε με την βάση δεδομένων και για να εκτελέσουμε queries.
- Επίσης , χρησιμοποιούμε και την βιβλιοθήκη random(την οποία πάλι μπορούμε να κατεβάσουμε με την βοήθεια του πακέτου pip) για την τυχαία επιλογή των δεδομένωνα από την βάση.
- Για την υλοποίηση της κλήρωσης αρχικά διαλέγουμε τυχαία συνταγές με την βοήθεια της συνάρτησης get_random_recipes η οποία επιλέγει τυχάια συνταγές , που δεν ανήκουν σε ένα σύνολο συνταγών(αυτές οι οποίες έχουν χρησιμοποιηθεί τα τελευταία 3 επεισόδεια και συνταγές που ανήκουν σε εθνική κουζίνα που έχει ήδη επιλεγεί για το επεισόδιο για το οποίο κάνουμε την επιλογή).
- Ομοίως, με την συνάρτηση get_chef_national επιλέγουμε τυχαία 1 chef για την συνταγή που έχουμε επιλέξει , ο οποίος έχει ειδικότητα σε αυτή την εθνική κουζίνα και δεν ανήκει σε ένα σύνολο πάλι(σεφ που έχουν επιλεγεί τα και τα 3 τελευταία επεισόδεια ή σεφ που έχουν ήδη επιλεγεί σε αυτό το επεισόδειο).
- Με παρόμοιο τρόπο επιλέγουμε και τους κριτές του συγκεκριμένου επεισοδείου.
- Ανάλογα με τις επιλογές μας έχουμε μια επιπλέον συνάρτηση που ενημερώνει τον αριθμό των εμφανίσεων όσο αφορά τους μάγειρες και τις συνταγές.

```
  GNU nano 7.2
random_script2.py
import psycopg2
import random

# Establish a connection to the database
conn = psycopg2.connect(
    dbname="masterchef",
    user="postgres",
    password="postgres",
    host="localhost",
    port="5432"
)
cursor = conn.cursor()

def get_random_recipes(cursor, num_recipes, excluded_recipes):
    national_cuisine_ids = []
    while len(national_cuisine_ids) < num_recipes:
        if excluded_recipes:
            cursor.execute("SELECT id_national FROM recipes WHERE id_rec NOT IN %s ORDER BY R
        else:
            cursor.execute("SELECT id_national FROM recipes ORDER BY RANDOM() LIMIT 1"
```

```
        id_national = cursor.fetchone()[0]
        if id_national not in national_cuisine_ids:
            national_cuisine_ids.append(id_national)
    return national_cuisine_ids

def get_chef_for_national(cursor, id_national, excluded_chefs, selected_chefs):
    excluded = tuple(excluded_chefs + selected_chefs)
    if excluded:
        cursor.execute("SELECT id_chef FROM chef_national WHERE id_national = %s AND id_chef
    else:
        cursor.execute("SELECT id_chef FROM chef_national WHERE id_national = %s ORDER BY RAN
    return cursor.fetchone()[0]

def get_judges(cursor, num_judges, excluded_chefs):
    cursor.execute("SELECT chef_id FROM chefs WHERE chef_id NOT IN %s ORDER BY RANDOM() LIMIT
    return [row[0] for row in cursor.fetchall()]

def update_appearance_counts(appearance_counts, selected_ids, max_appearances):
    for key in selected_ids:
        if key in appearance_counts:
            appearance_counts[key] += 1
        else:
            appearance_counts[key] = 1

    # Remove any items that exceed max_appearances
    return {key: count for key, count in appearance_counts.items() if count < max_ap

def populate_episodes(cursor):
    cursor.execute("SELECT no_ep FROM episodes")
    episodes = cursor.fetchall()

    national_appearance_counts = {}
    chef_appearance_counts = {}
    judge_appearance_counts = {}
    recipe_appearance_counts = {}

    for episode in episodes:
        episode_id = episode[0]

        # Step 1: Get 10 distinct national cuisine IDs
        excluded_recipes = [rec_id for rec_id, count in recipe_appearance_counts.ite      if
        national_cuisine_ids = get_random_recipes(cursor, 10, excluded_recipes)

        # Step 2: Select one chef for each national cuisine ID, ensuring no duplicates within
        chefs = []
        selected_chefs = []
        for id_national in national_cuisine_ids:
            excluded_chefs = [chef_id for chef_id, count in chef_appearance_counts.i        if
            chef_id = get_chef_for_national(cursor, id_national, excluded_chefs, sel
            if chef_id in selected_chefs:
                continue
            chefs.append((chef_id, episode_id, id_national))
            selected_chefs.append(chef_id)

        # Step 3: Select 3 judges (chefs not in the previous list)
        excluded_judges = [chef_id for chef_id, count in judge_appearance_counts.ite      if
```

```
        judges = get_judges(cursor, 3, excluded_judges)

        # Update appearance counts
        national_appearance_counts = update_appearance_counts(national_appearance_co
        chef_appearance_counts = update_appearance_counts(chef_appearance_counts, se
        judge_appearance_counts = update_appearance_counts(judge_appearance_counts,
        recipe_ids = [rec_id for rec_id in excluded_recipes if rec_id in [id_nationa  for
        recipe_appearance_counts = update_appearance_counts(recipe_appearance_counts

        # Insert into chefs_episodes
        for chef_id, episode_id, id_national in chefs:
            cursor.execute(
                "INSERT INTO chefs_episodes (chef_id, chef_no_ep, id_national, rec_id) VALUES
                (chef_id, episode_id, id_national, id_national)
            )

        # Insert into judge_episodes
        for judge_id in judges:
            cursor.execute(
                "INSERT INTO judge_episodes (judge_id, judge_no_ep) VALUES (%s, %s)"
                (judge_id, episode_id)
            )

# Populate the tables
populate_episodes(cursor)

# Commit the transaction
conn.commit()

# Close the connection
cursor.close()
conn.close()
```

Τέλος θα τρέξουμε ακόμη ένα script για να κάνουμε populate τον πίνακα score όπου αποθηκεύουμε τις βαθμολογίες των 3 κριτών που έχουμε επιλέξει για ένα επεισόδιο για κάθε διαγωνιζόμενο σεφ του επισοδείου.

- Και σε αυτή την περίπτωση χρησιμοποιούμε την βιβλιοθήκη psycop2 και random για να ενωθούμε και να τρέουμε queries για την βάση αλλά και για να αναυθέσουμε βαθμολογία με ένα τυχαίο τρόπο από κάθε κριτή σε κάθε διαγωνιζόμενο για κάθε επεισόδειο.
- Έχουμε μια συνάρτηση "get_random_score" η οποία μας επιστρέφει ένα αριθμό από το 1 μέχρι το 5 με πιθανότητα 0.125 για το 1 ή το 5 και 0.25 για το 2,3,4.Με αυτό τον τρόπο θεωρήσαμε ότι είναι λιγότερο πιθανό να έχουμε μια τέλεια βαθμολογία ή την χειρότερη.
- Επίσης έχουμε μια συνάρτηση populate_score η οποία με παρόμοιο τρόπο με πρίν , με την βοήθεια της βιβλιοθήκης psycopg2 κάνει populate τον πίνακα score.
- Πιο συγκεκριμένα για κάθε tuple του πίνακα chef_episode(1η λούπα) δίνουμε ένα score (τυχαία) από κάθε μάγειρα κριτή(δεύτερη λούπα)

```
  GNU nano 7.2                                          populate_score.py
import psycopg2
import random

# Establish a connection to the database
conn = psycopg2.connect(
    dbname="masterchef",
    user="postgres",
    password="postgres",
    host="localhost",
    port="5432"
)
cursor = conn.cursor()

def get_random_score():
    # Define the scores with the desired probabilities
    scores = [1, 2, 3, 4, 5]
    probabilities = [0.125, 0.25, 0.25, 0.25, 0.125]
    return random.choices(scores, probabilities)[0]

def populate_scores(cursor):
    # Fetch all chefs episodes
    cursor.execute("SELECT chef_id, chef_no_ep FROM chefs_episodes")
    chefs_episodes = cursor.fetchall()

    for chef_id, chef_no_ep in chefs_episodes:
        # Fetch judges for the same episode
        cursor.execute("SELECT judge_id, judge_no_ep FROM judge_episodes WHERE judge_no_ep =
        judges = cursor.fetchall()

        for judge_id, judge_no_ep in judges:
            score = get_random_score()
            cursor.execute(
                "INSERT INTO score (chef_id, chef_no_ep, judge_id, judge_no_ep, score) VALUES
                (chef_id, chef_no_ep, judge_id, judge_no_ep, score)
            )

# Populate the score table
populate_scores(cursor)

# Commit the transaction
conn.commit()

# Close the connection
cursor.close()
conn.close()
```

8 Προσθέτουμε τις εικόνες

Για την εισαγωγή των εικονών στους πίνακες recipes,ingredients,equipment έχουμε αποθηκεύσει τις εικόνες από το διαδύκτυο σε μορφή png και με όνομα το id τους στον συγκεκριμένο πίνκα στα directories ingredients_images,recipes_images και equipment_images, ενώ για τις εικόνες για τους chef έχουμε γράψει ένα script στην python που αυτοματοποιεί αυτή ακριβώς την διαδικασία.

Αρχικά τρέχουμε το script generate_chef_images.py που δημιουργεί για κάθε μάγειρα μια εικόνα με την χρήση ενός online random genrator.

- Το script αυτό χρησιμοποιεί επιπλέον τις βιβλιοθήκες requests,PIL και io, ενώ και πάλι χρησιμοποιούμε την os για να ορίσουμε το directory που θέλουμε να κάνουμε generate τις εικόνες
- Αρικά δημιουργούμε το directory στο οποίο θέλουμε να αποθηκεύσουμε τις εικόνες με την βοήθεια της os.
- Στη συνέχεια όρίζουμε την ιστοσελίδα(url) από την οποία θα κάνουμε generate τις εικόνες.
- Ακολούθως ορίζουμε μια συνάρτηση (download_image)η οποία θα είναι υπεύθυνη για να κατεβάζει εικόνες από το url που θέσαμε προηγουμένως με την βοήθεια ενός API από την βιβλιοθήκη PIL.
- Στην συνέχεια ορίζουμε σε μια λίστα όλους τους chef , με το όνομά τους το φύλο τους και την ηλικία τους για να έχουμε πιο ρεαλιστικές εικόνες.
- Επίσης, έχουμε και την συνάρτηση add_text_to_image η οποία προσθέτει κείμενο στην εικόνα(θα προσθέσουμε το όνοματεπώνυμο και την ηλικία του σεφ)
- Τέλος, για κάθε chef δημιουργούμε μια εικόνα από το random online generator , την παίρνουμε μέσω του API,της προσθέτουμε κείμενο με την βοήθεια της προηγούμενης συνάρτησης και την αποθηκεύουμε στο directory που έχουμε ορίσει για output.

```
  GNU nano 7.2
generate_chef_images.py
import os
import requests
from PIL import Image, ImageDraw, ImageFont
from io import BytesIO

# Ensure the directory exists
output_dir = '/home/georgeandro/george_home/semfe/8o/database/ergasia/git_repo_masterchef/che
os.makedirs(output_dir, exist_ok=True)

face_url = "https://randomuser.me/api/portraits/"

def download_image(url, save_path):
    response = requests.get(url)
    response.raise_for_status()
    img = Image.open(BytesIO(response.content))
    img = img.convert("RGB")  # Ensure the image is in RGB mode
    img.save(save_path, 'PNG')
```

```
# Data for chefs with their sex and age
chefs = [
    (1, 'Kylie', 'female', 49), (2, 'Milty', 'male', 54), (3, 'Drucy', 'female', 49)
    (4, 'Mandel', 'male', 30), (5, 'Ginelle', 'female', 45), (6, 'Rosalind', 'female'
    (7, 'Hailee', 'female', 58), (8, 'Gannon', 'male', 44), (9, 'Jerrome', 'male', 2
    (10, 'Zandra', 'female', 54), (11, 'Godard', 'male', 44), (12, 'Gerty', 'female'
    (13, 'Willy', 'male', 29), (14, 'Wren', 'female', 51), (15, 'Zulema', 'female',
    (16, 'Lem', 'male', 31), (17, 'Marjie', 'female', 59), (18, 'Dela', 'female', 48
    (19, 'Kassandra', 'female', 26), (20, 'Ferdinande', 'female', 53), (21, 'Nickolaus'   'mal
    (22, 'Donica', 'female', 46), (23, 'Joyann', 'female', 41), (24, 'Fidela', 'female'
    (25, 'Irene', 'female', 58), (26, 'Guido', 'male', 37), (27, 'Roch', 'male', 40)
    (28, 'Kinnie', 'female', 39), (29, 'Dulcia', 'female', 56), (30, 'Quinton', 'male'
    (31, 'Aubrette', 'female', 38), (32, 'Rodger', 'male', 49), (33, 'Tansy', 'female'
    (34, 'Mae', 'female', 37), (35, 'Andris', 'male', 23), (36, 'Jarib', 'male', 23)
    (37, 'Albertina', 'female', 55), (38, 'Colline', 'female', 38), (39, 'Benny', 'male'
    (40, 'Balduin', 'male', 28), (41, 'Paule', 'female', 32), (42, 'Rozalin', 'female'
    (43, 'Lindi', 'female', 34), (44, 'Angelico', 'male', 37), (45, 'Howard', 'male'
    (46, 'Igor', 'male', 34), (47, 'Benni', 'male', 38), (48, 'Mervin', 'male', 49),
    (49, 'Haleigh', 'female', 57), (50, 'Samuele', 'male', 23)
]

def add_text_to_image(image, text, position):
    draw = ImageDraw.Draw(image)
    font = ImageFont.load_default()
    draw.text(position, text, fill="white", font=font)

for chef in chefs:
    chef_id, name, sex, age = chef
    gender = 'men' if sex == 'male' else 'women'
    image_id = chef_id % 100  # RandomUser API has 100 images for each gender
    image_url = f"{face_url}/{gender}/{image_id}.jpg"
    save_path = os.path.join(output_dir, f"{chef_id}.png")
    try:
        download_image(image_url, save_path)
        img = Image.open(save_path)
        add_text_to_image(img, f"Name: {name}", (10, 10))
        add_text_to_image(img, f"Sex: {sex}", (10, 30))
        add_text_to_image(img, f"Age: {age}", (10, 50))
        img.save(save_path, 'PNG')
        print(f"Saved image {chef_id}.png with details")
    except Exception as e:
        print(f"Error downloading {image_url}: {e}")

print("Downloaded 50 random faces with details.")
```

Πλέον αφού έχουμε όλες της απαραίτητες εικόνες αποθηκευμένες στα σχετικά directory θα τρέξουμε τα script python equipment_images.py,chefs_images_populate.py,ingredients_images.py και recipes_images.py για να προσθέσουμε τις εικόνες στα σχετικά tables.Προσοχή!!!Πρώτα πρέπει να τρέξουμε το script create_image_columns.sql για να δημιουργήσουμε με την βοήθεια της εντολής ALTER TABLE τις αντοίστοιχες στήλες σε αυτά τα tables, με τύπο BYTEA.
Πιο κάτω θα εξηγήσουμε μόνο ένα παό τα python scripts αφού κάνουν την ίδια δουλειά.

- Χρησιμοποιούμε τις βιβλιοθήκες os και psycopg2 για να ενωθούμε με την βάση και να πάρουμε τα δεδομένα από τα αντίστοιχα directories.

- Προσοχή όταν τρέξετε τα συγκεκριμένα scripts να αλλάξετε τα σχετικά directories αλλά και options της βάσης για τον δικό σας υπολογιστή.
- Η συνάρτηση upload_images κάνει διαπέραση όλων των αρχείων του directory με τις εικόνες και για το tuple που έχει το ίδιο id με το όνομα του png κάνουμε Update table προσθέτοντας την εικόνα

pythonsytle

```
import os
import psycopg2

# Database connection parameters
conn_params = {
    'dbname': 'masterchef',
    'user': 'postgres',
    'password': 'postgres',
    'host': 'localhost',
    'port': '5432'
}

# Directory where images are stored
image_directory = '/home/georgeandro/george_home/semfe/8o/database/ergasia/git_repo_masterche

def upload_images(conn, image_dir):
    cursor = conn.cursor()

    # Loop through all files in the image directory
    for filename in os.listdir(image_dir):
        # Assuming filenames are like '1.png', '2.png', etc., corresponding to ing_id
        ing_id, ext = os.path.splitext(filename)
        if ext.lower() == '.png' and ing_id.isdigit():
            ing_id = int(ing_id)
            image_path = os.path.join(image_dir, filename)

            # Read the image file
            with open(image_path, 'rb') as file:
                image_data = file.read()

            # Update the image column for the corresponding ing_id
            cursor.execute("""
                UPDATE ingredients
                SET image = %s
                WHERE ing_id = %s
            """, (psycopg2.Binary(image_data), ing_id))

    conn.commit()
    cursor.close()

if __name__ == '__main__':
    try:
        # Connect to the PostgreSQL database
        conn = psycopg2.connect(**conn_params)
        upload_images(conn, image_directory)
    except Exception as e:
        print(f"Error: {e}")
```

```
    finally:
        if conn:
            conn.close()
```

9 Queries

Ήρθε η ώρα να απαντήσουμε στα ερωτήματα που ζητάει η εργασία αφού η βάση βρίσκεται στην τελική της μορφή.

3.1. Μέσος Όρος Αξιολογήσεων (σκορ) ανά μάγειρα και Εθνική κουζίνα.

```
SELECT
    nc.name AS cuisine_type,
    c.chef_name,
    c.surname,
    AVG(s.score) AS average_score
FROM
    score s
JOIN
    chefs_episodes ce ON s.chef_id = ce.chef_id AND s.chef_no_ep = ce.chef_no_ep
JOIN
    national_cuisine nc ON ce.id_national = nc.id_national
JOIN
    chefs c ON s.chef_id = c.chef_id
GROUP BY
    nc.name, c.chef_name, c.surname
ORDER BY
    c.chef_name, c.surname;
```

Αποτελέσματα Query(LIMIT 10 για οικονομία χώρου):

```
 cuisine_type | chef_name | surname |    average_score
--------------+-----------+---------+--------------------
 Hungary      | Albertina | Axel    | 3.0000000000000000
 Japan        | Albertina | Axel    | 2.8333333333333333
 Italy        | Albertina | Axel    | 2.6666666666666667
 India        | Albertina | Axel    | 3.6666666666666667
 India        | Andris    | Ganning | 2.0000000000000000
 Morocco      | Andris    | Ganning | 2.3333333333333333
 Italy        | Andris    | Ganning | 2.8333333333333333
 Korea        | Andris    | Ganning | 3.0000000000000000
 Croatia      | Andris    | Ganning | 3.6666666666666667
 China        | Andris    | Ganning | 2.5000000000000000
(10 rows)
```

3.2. Για δεδομένη Εθνική κουζίνα και έτος, ποιοι μάγειρες ανήκουν σε αυτήν και ποιοι μάγειρες συμμετείχαν σε επεισόδια;

```
WITH EpisodeRanges AS (
    SELECT 1 AS year, 1 AS start_ep, 10 AS end_ep UNION ALL
    SELECT 2 AS year, 11 AS start_ep, 20 AS end_ep UNION ALL
    SELECT 3 AS year, 21 AS start_ep, 30 AS end_ep UNION ALL
    SELECT 4 AS year, 31 AS start_ep, 40 AS end_ep UNION ALL
    SELECT 5 AS year, 41 AS start_ep, 50 AS end_ep
)
SELECT
    chefs.chef_name,
    chefs.surname,
    national_cuisine.name AS national_cuisine,
    er.year,
    CASE
        WHEN ce.chef_id IS NOT NULL THEN 'Participated'
        ELSE 'Not Participated'
    END AS participation_status
FROM
    chefs
JOIN
    chef_national ON chefs.chef_id = chef_national.id_chef
JOIN
    national_cuisine ON chef_national.id_national = national_cuisine.id_national
LEFT JOIN
    chefs_episodes ce ON chefs.chef_id = ce.chef_id
LEFT JOIN
    episodes e ON ce.chef_no_ep = e.no_ep
JOIN
    EpisodeRanges er ON e.no_ep BETWEEN er.start_ep AND er.end_ep
WHERE
    national_cuisine.name LIKE '%United States%'
    AND er.year = 2
    ORDER BY
    chefs.surname;
```

Αποτελέσματα Query(Για δευτερη σεζον και για ΗΠΑ):

```
 chef_name |   surname    | national_cuisine | year | participation_status
-----------+--------------+------------------+------+----------------------
 Irene     | Barenskie    | United States    |    2 | Participated
 Irene     | Barenskie    | United States    |    2 | Participated
 Lem       | Boyall       | United States    |    2 | Participated
 Lem       | Boyall       | United States    |    2 | Participated
 Angelico  | Bubear       | United States    |    2 | Participated
 Angelico  | Bubear       | United States    |    2 | Participated
 Quinton   | Cottey       | United States    |    2 | Participated
 Quinton   | Cottey       | United States    |    2 | Participated
 Ginelle   | Glasscoe     | United States    |    2 | Participated
 Ginelle   | Glasscoe     | United States    |    2 | Participated
 Ginelle   | Glasscoe     | United States    |    2 | Participated
 Nickolaus | Goldsworthy  | United States    |    2 | Participated
 Dulcia    | Ibel         | United States    |    2 | Participated
 Dulcia    | Ibel         | United States    |    2 | Participated
 Hailee    | Joddins      | United States    |    2 | Participated
```

```
 Hailee    | Joddins      | United States    |    2 | Participated
 Hailee    | Joddins      | United States    |    2 | Participated
 Marjie    | Kidd         | United States    |    2 | Participated
 Zulema    | Mufford      | United States    |    2 | Participated
 Zulema    | Mufford      | United States    |    2 | Participated
 Zulema    | Mufford      | United States    |    2 | Participated
 Mae       | O' Cuolahan  | United States    |    2 | Participated
 Paule     | Peddersen    | United States    |    2 | Participated
 Paule     | Peddersen    | United States    |    2 | Participated
 Balduin   | Rolfs        | United States    |    2 | Participated
 Balduin   | Rolfs        | United States    |    2 | Participated
 Igor      | Stafford     | United States    |    2 | Participated
 Mervin    | Thaller      | United States    |    2 | Participated
 Lindi     | Wermerling   | United States    |    2 | Participated
 Lindi     | Wermerling   | United States    |    2 | Participated
(30 rows)
```

3.3 Βρείτε τους νέους μάγειρες (ηλικία < 30 ετών) που έχουν τις περισσότερες συνταγές.

```
SELECT
    chef_name,
    surname,
    chef_age,
    recipe_count
FROM (
    SELECT
        c.chef_name,
        c.surname,
        c.chef_age,
        (SELECT COUNT(*)
         FROM chefs_episodes ce
         WHERE ce.chef_id = c.chef_id AND ce.rec_id IN (
             SELECT r.id_rec
             FROM recipes r
             WHERE r.id_rec = ce.rec_id)) AS recipe_count
    FROM
        chefs c
    WHERE
        c.chef_age < 30
) AS chef_recipes
ORDER BY
    recipe_count DESC;
```

Αποτελέσματα Query:

```
 chef_name |  surname   | chef_age | recipe_count
-----------+------------+----------+--------------
 Howard    | Wakes      |       26 |           17
 Samuele   | Swannick   |       23 |           16
 Fidela    | Wozencraft |       29 |           16
 Tansy     | Southward  |       26 |           13
 Willy     | Lere       |       29 |           13
 Andris    | Ganning    |       23 |           11
 Jerrome   | Gotcliffe  |       23 |           11
 Benny     | Fraine     |       20 |           11
 Rozalin   | Foss       |       28 |           11
 Balduin   | Rolfs      |       28 |           10
 Zulema    | Mufford    |       25 |            9
 Jarib     | Varga      |       23 |            8
 Kassandra | Harkes     |       26 |            5
(13 rows)
```

3.4 Βρείτε τους μάγειρες που δεν έχουν συμμετάσχει ποτέ σε ως κριτές σε κάποιο επεισόδιο.

```
SELECT
    chef_name,
    surname
FROM
    chefs
WHERE
    chef_id NOT IN (
        SELECT judge_id
        FROM judge_episodes
    );
```

Αποτελέσματα Query:

```
 chef_name | surname
-----------+---------
 Dulcia    | Ibel
(1 row)
```

3.5 Ποιοι κριτές έχουν συμμετάσχει στον ίδιο αριθμό επεισοδίων σε διάστημα ενός έτους με περισσότερες από 3 εμφανίσεις;

```
WITH JudgeAppearances AS (
    SELECT
        je.judge_id,
        c.chef_name,
        c.surname,
        CASE
            WHEN je.judge_no_ep BETWEEN 1 AND 10 THEN 1
            WHEN je.judge_no_ep BETWEEN 11 AND 20 THEN 2
            WHEN je.judge_no_ep BETWEEN 21 AND 30 THEN 3
            WHEN je.judge_no_ep BETWEEN 31 AND 40 THEN 4
            WHEN je.judge_no_ep BETWEEN 41 AND 50 THEN 5
        END AS year,
        COUNT(je.judge_no_ep) AS appearances
    FROM
        judge_episodes je
    JOIN
        chefs c ON je.judge_id = c.chef_id
    WHERE
        je.judge_no_ep BETWEEN 1 AND 50
    GROUP BY
        je.judge_id, c.chef_name, c.surname, year
    HAVING
        COUNT(je.judge_no_ep) > 3
)
SELECT
    chef_name,
    surname,
    appearances,
    year
FROM
    JudgeAppearances
GROUP BY
    appearances, year, chef_name, surname
ORDER BY
   appearances DESC, year;
```

Αποτελέσματα Query:

```
 chef_name |   surname    | appearances | year
-----------+--------------+-------------+------
 Mae       | O' Cuolahan |           4 |    2
(1 row)
```

3.6 Βρείτε τα 3 κορυφαία (top-3) ζεύγη που εμφανίστηκαν σε επεισόδια

```
EXPLAIN ANALYZE
WITH RecipeTags AS (
    SELECT
        tr1.id_rec,
        t1.name AS tag1,
        t2.name AS tag2
    FROM
        tags_rec tr1
    JOIN
        tags t1 ON tr1.id_tags = t1.id_tags
    JOIN
        tags_rec tr2 ON tr1.id_rec = tr2.id_rec
    JOIN
        tags t2 ON tr2.id_tags = t2.id_tags
    WHERE
        t1.id_tags < t2.id_tags
),
TagPairsCount AS (
    SELECT
        rt.tag1,
        rt.tag2,
        COUNT(*) AS pair_count
    FROM
        RecipeTags rt
    GROUP BY
        rt.tag1, rt.tag2
)
SELECT
    tpc.tag1,
    tpc.tag2,
    tpc.pair_count
FROM
    TagPairsCount tpc
ORDER BY
    tpc.pair_count DESC
LIMIT 3;
```

Αποτελέσματα Query:

```
  tag1   |     tag2      | pair_count
---------+---------------+------------
 Dessert | Baked Goods |           6
 Seafood | Holiday     |           5
 Seafood | Dessert     |           5
(3 rows)
```

Στη συνέχει θα προσθέτουμε τα παρακάτω indexes(δημιουργούμε και διαγράφουμε τα indexes με τις παρακάτω εντολές):

```
--create the index for 3.6)
CREATE INDEX idx_tags_rec_id_rec ON tags_rec (id_rec);
CREATE INDEX idx_tags_rec_id_tags ON tags_rec (id_tags);

--drop indexes for 3.6)
```

```
DROP INDEX IF EXISTS idx_tags_rec_id_rec;
DROP INDEX IF EXISTS idx_tags_rec_id_tags;
```

Query Performance με και χωρίς τα indexes:

```
--before indexing
Limit  (cost=88.57..88.58 rows=3 width=244) (actual time=2.375..2.380 rows=3 loops=1
  ->  Sort  (cost=88.57..89.79 rows=487 width=244) (actual time=2.373..2.377 rows=3
        Sort Key: (count(*)) DESC
        Sort Method: top-N heapsort  Memory: 25kB
        ->  HashAggregate  (cost=72.54..77.41 rows=487 width=244) (actual time=2.117
              Group Key: t1.name, t2.name
              Batches: 1  Memory Usage: 105kB
              ->  Hash Join  (cost=25.65..68.88 rows=487 width=236) (actual time=0.6
                    Hash Cond: (tr1.id_rec = tr2.id_rec)
                    Join Filter: (t1.id_tags < t2.id_tags)
                    Rows Removed by Join Filter: 865
                    ->  Nested Loop  (cost=0.16..22.11 rows=270 width=126) (actual t
                          ->  Seq Scan on tags_rec tr1  (cost=0.00..4.70 rows=270 wi
                          ->  Memoize  (cost=0.16..0.36 rows=1 width=122) (actual ti
                                Cache Key: tr1.id_tags
                                Cache Mode: logical
                                Hits: 240  Misses: 30  Evictions: 0  Overflows: 0  M
                                ->  Index Scan using tags_pkey on tags t1  (cost=0.1
                                      Index Cond: (id_tags = tr1.id_tags)
                    ->  Hash  (cost=22.11..22.11 rows=270 width=126) (actual time=0
                          Buckets: 1024  Batches: 1  Memory Usage: 22kB
                          ->  Nested Loop  (cost=0.16..22.11 rows=270 width=126) (ac
                                ->  Seq Scan on tags_rec tr2  (cost=0.00..4.70 rows=
                                ->  Memoize  (cost=0.16..0.36 rows=1 width=122) (act
                                      Cache Key: tr2.id_tags
                                      Cache Mode: logical
                                      Hits: 240  Misses: 30  Evictions: 0  Overflows
Memory Usage: 4kB
                                      ->  Index Scan using tags_pkey on tags t2  (co
                                            Index Cond: (id_tags = tr2.id_tags)
Planning Time: 1.287 ms
Execution Time: 2.475 ms



--after indexing

Limit  (cost=88.57..88.58 rows=3 width=244) (actual time=1.030..1.034 rows=3 loops=1
  ->  Sort  (cost=88.57..89.79 rows=487 width=244) (actual time=1.029..1.032 rows=3
        Sort Key: (count(*)) DESC
        Sort Method: top-N heapsort  Memory: 25kB
        ->  HashAggregate  (cost=72.54..77.41 rows=487 width=244) (actual time=0.897
              Group Key: t1.name, t2.name
              Batches: 1  Memory Usage: 105kB
              ->  Hash Join  (cost=25.65..68.88 rows=487 width=236) (actual time=0.2
                    Hash Cond: (tr1.id_rec = tr2.id_rec)
                    Join Filter: (t1.id_tags < t2.id_tags)
                    Rows Removed by Join Filter: 865
                    ->  Nested Loop  (cost=0.16..22.11 rows=270 width=126) (actual t
                          ->  Seq Scan on tags_rec tr1  (cost=0.00..4.70 rows=270 wi
                          ->  Memoize  (cost=0.16..0.36 rows=1 width=122) (actual ti
```

```
                                Cache Key: tr1.id_tags
                                Cache Mode: logical
                                Hits: 240  Misses: 30  Evictions: 0  Overflows: 0  M
                                ->  Index Scan using tags_pkey on tags t1  (cost=0.1
                                      Index Cond: (id_tags = tr1.id_tags)
                      ->  Hash  (cost=22.11..22.11 rows=270 width=126) (actual time=0
                            Buckets: 1024  Batches: 1  Memory Usage: 22kB
                            ->  Nested Loop  (cost=0.16..22.11 rows=270 width=126) (ac
                                  ->  Seq Scan on tags_rec tr2  (cost=0.00..4.70 rows=
                                  ->  Memoize  (cost=0.16..0.36 rows=1 width=122) (act
                                        Cache Key: tr2.id_tags
                                        Cache Mode: logical
                                        Hits: 240  Misses: 30  Evictions: 0  Overflows
Memory Usage: 4kB
                                        ->  Index Scan using tags_pkey on tags t2  (co
                                              Index Cond: (id_tags = tr2.id_tags)
Planning Time: 0.723 ms
Execution Time: 1.091 ms
```

Παρατηρούμε ότι ήταν αποτελεσματικό να προσθέσουμε indexes στον πίνακα tags_rec αφού ο χρόνος εκτέλεσης για το query έπεσε από $2.475ms$ σε $1.091ms$

3.7. Βρείτε όλους τους μάγειρες που συμμετείχαν τουλάχιστον 5 λιγότερες φορές από τον μάγειρα με τις περισσότερες συμμετοχές σε επεισόδια.

```
WITH ChefAppearances AS (
    SELECT
        c.chef_id,
        c.chef_name,
        c.surname,
        COUNT(DISTINCT ce.chef_no_ep) AS chef_appearances,
        COUNT(DISTINCT je.judge_no_ep) AS judge_appearances
    FROM
        chefs c
    LEFT JOIN
        chefs_episodes ce ON c.chef_id = ce.chef_id
    LEFT JOIN
        judge_episodes je ON c.chef_id = je.judge_id
    GROUP BY
        c.chef_id, c.chef_name, c.surname
),
TotalAppearances AS (
    SELECT
        ca.chef_id,
        ca.chef_name,
        ca.surname,
        ca.chef_appearances,
        ca.judge_appearances,
        ca.chef_appearances + ca.judge_appearances AS total_appearances
    FROM
        ChefAppearances ca
),
MaxAppearances AS (
    SELECT
        MAX(total_appearances) AS max_appearances
    FROM
        TotalAppearances
)
SELECT
    ta.chef_name,
    ta.surname,
    ta.chef_appearances,
    ta.judge_appearances,
    ta.total_appearances
FROM
    TotalAppearances ta,
    MaxAppearances ma
WHERE
    ta.total_appearances <= ma.max_appearances - 5
ORDER BY
    ta.total_appearances DESC;
```

Αποτελέσματα query:

```
 chef_name   |    surname     | chef_appearances | judge_appearances | total_appearanc
-------------+----------------+------------------+-------------------+-----------------
 Mae         | O' Cuolahan    |               11 |                 6 |              17
```

Paule	Peddersen	13	4	17
Samuele	Swannick	16	1	17
Fidela	Wozencraft	16	1	17
Jerrome	Gotcliffe	11	5	16
Ginelle	Glasscoe	15	1	16
Dela	Christal	12	4	16
Mandel	Kitcat	14	1	15
Angelico	Bubear	12	3	15
Rozalin	Foss	11	4	15
Quinton	Cottey	11	4	15
Ferdinande	Bubeer	10	5	15
Jarib	Varga	8	6	14
Zandra	Fitzhenry	11	3	14
Roch	Kopisch	10	4	14
Andris	Ganning	11	3	14
Balduin	Rolfs	10	4	14
Hailee	Joddins	13	1	14
Dulcia	Ibel	14	0	14
Mervin	Thaller	8	5	13
Irene	Barenskie	11	1	12
Benny	Fraine	11	1	12
Albertina	Axel	6	6	12
Drucy	Atmore	8	3	11
Guido	Behrendsen	9	2	11
Colline	Beldon	6	5	11
Gerty	Stacy	7	4	11
Zulema	Mufford	9	2	11
Nickolaus	Goldsworthy	6	4	10
Joyann	Brimley	8	2	10
Kinnie	Orto	8	2	10
Igor	Stafford	7	3	10
Aubrette	Habershaw	5	4	9
Lem	Boyall	8	1	9
Milty	Shufflebotham	7	2	9
Rodger	Bottomore	4	5	9
Donica	Baulcombe	7	1	8
Lindi	Wermerling	3	5	8
Godard	Chagg	7	1	8
Rosalind	Dudman	5	2	7
Kassandra	Harkes	5	2	7
Benni	Penrose	5	1	6
Kylie	Gadney	3	1	4

3.8. Σε ποιο επεισόδιο χρησιμοποιήθηκαν τα περισσότερα εξαρτήματα (εξοπλισμός);

```
EXPLAIN ANALYZE
SELECT
    e.no_ep,
    COUNT(re.equipment_id) AS equipment_count
FROM
    episodes e
JOIN
    chefs_episodes ce ON e.no_ep = ce.chef_no_ep
JOIN
    recipes r ON ce.rec_id = r.id_rec
JOIN
    recipes_has_equipment re ON r.id_rec = re.recipes_id
GROUP BY
    e.no_ep
ORDER BY
    equipment_count DESC
LIMIT 1;
```

Αποτελέσματα Query:

```
 no_ep | equipment_count
-------+-----------------
    35 |              65
(1 row)
```

Στη συνέχει θα προσθέτουμε τα παρακάτω indexes(δημιουργούμε και διαγράφουμε τα indexes με τις παρακάτω εντολές):

```
--3.8)
--dimiourgoume indexes
CREATE INDEX idx_chefs_episodes_no_ep ON chefs_episodes (chef_no_ep);
CREATE INDEX idx_recipes_id_rec ON recipes (id_rec);
CREATE INDEX idx_recipes_has_equipment_recipes_id ON recipes_has_equipment (recipes_
--drop indexes
DROP INDEX IF EXISTS idx_chefs_episodes_no_ep;
DROP INDEX IF EXISTS idx_recipes_has_equipment_recipes_id;
```

Query Performance με και χωρίς τα indexes:

```
--non indexed results
Limit  (cost=142.87..142.87 rows=1 width=12) (actual time=2.909..2.912 rows=1 loops=
  ->  Sort  (cost=142.87..149.24 rows=2550 width=12) (actual time=2.907..2.909 rows=
        Sort Key: (count(re.equipment_id)) DESC
        Sort Method: top-N heapsort  Memory: 25kB
        ->  HashAggregate  (cost=104.62..130.12 rows=2550 width=12) (actual time=2.8
              Group Key: e.no_ep
              Batches: 1  Memory Usage: 121kB
              ->  Hash Join  (cost=25.62..91.08 rows=2708 width=8) (actual time=0.56
                    Hash Cond: (ce.rec_id = r.id_rec)
                    ->  Nested Loop  (cost=0.17..34.16 rows=500 width=8) (actual tim
                          ->  Seq Scan on chefs_episodes ce  (cost=0.00..8.00 rows=5
                          ->  Memoize  (cost=0.17..0.28 rows=1 width=4) (actual time
                                Cache Key: ce.chef_no_ep
                                Cache Mode: logical
```

```
                                Hits: 450  Misses: 50  Evictions: 0  Overflows: 0  M
                                ->  Index Only Scan using episodes_pkey on episodes
(cost=0.15..0.27 rows=1 width=4) (actual time=0.002..0.002 rows=1 loops=50)
                                      Index Cond: (no_ep = ce.chef_no_ep)
                                      Heap Fetches: 50
                    ->  Hash  (cost=21.87..21.87 rows=287 width=12) (actual time=0.5
                          Buckets: 1024  Batches: 1  Memory Usage: 21kB
                          ->  Hash Join  (cost=16.19..21.87 rows=287 width=12) (actu
                                Hash Cond: (re.recipes_id = r.id_rec)
                                ->  Seq Scan on recipes_has_equipment re  (cost=0.00
                                ->  Hash  (cost=15.53..15.53 rows=53 width=4) (actua
                                      Buckets: 1024  Batches: 1  Memory Usage: 10kB
                                      ->  Seq Scan on recipes r  (cost=0.00..15.53 r
Planning Time: 1.721 ms
Execution Time: 3.035 ms

--indexed results
Limit  (cost=142.87..142.87 rows=1 width=12) (actual time=1.491..1.494 rows=1 loops=
  ->  Sort  (cost=142.87..149.24 rows=2550 width=12) (actual time=1.490..1.492 rows=
        Sort Key: (count(re.equipment_id)) DESC
        Sort Method: top-N heapsort  Memory: 25kB
        ->  HashAggregate  (cost=104.62..130.12 rows=2550 width=12) (actual time=1.4
              Group Key: e.no_ep
              Batches: 1  Memory Usage: 121kB
              ->  Hash Join  (cost=25.62..91.08 rows=2708 width=8) (actual time=0.21
                    Hash Cond: (ce.rec_id = r.id_rec)
                    ->  Nested Loop  (cost=0.17..34.16 rows=500 width=8) (actual tim
                          ->  Seq Scan on chefs_episodes ce  (cost=0.00..8.00 rows=5
                          ->  Memoize  (cost=0.17..0.28 rows=1 width=4) (actual time
                                Cache Key: ce.chef_no_ep
                                Cache Mode: logical
                                Hits: 450  Misses: 50  Evictions: 0  Overflows: 0  M
                                ->  Index Only Scan using episodes_pkey on episodes
(cost=0.15..0.27 rows=1 width=4) (actual time=0.001..0.001 rows=1 loops=50)
                                      Index Cond: (no_ep = ce.chef_no_ep)
                                      Heap Fetches: 50
                    ->  Hash  (cost=21.87..21.87 rows=287 width=12) (actual time=0.1
                          Buckets: 1024  Batches: 1  Memory Usage: 21kB
                          ->  Hash Join  (cost=16.19..21.87 rows=287 width=12) (actu
                                Hash Cond: (re.recipes_id = r.id_rec)
                                ->  Seq Scan on recipes_has_equipment re  (cost=0.00
                                ->  Hash  (cost=15.53..15.53 rows=53 width=4) (actua
                                      Buckets: 1024  Batches: 1  Memory Usage: 10kB
                                      ->  Seq Scan on recipes r  (cost=0.00..15.53 r
Planning Time: 0.798 ms
Execution Time: 1.560 ms
```

Παρατηρούμε ότι ήταν αποτελεσματικό να προσθέσουμε indexes στον πίνακα tags_rec αφού ο χρόνος εκτέλεσης για το query έπεσε από $3.035ms$ σε $1.56ms$

3.9. Λίστα με μέσο όρο αριθμού γραμμάριων υδατανθράκων στο διαγωνισμό ανά έτος;

```
WITH EpisodeYears AS (
    SELECT
        no_ep,
        CASE
            WHEN no_ep BETWEEN 1 AND 10 THEN 1
            WHEN no_ep BETWEEN 11 AND 20 THEN 2
            WHEN no_ep BETWEEN 21 AND 30 THEN 3
            WHEN no_ep BETWEEN 31 AND 40 THEN 4
            WHEN no_ep BETWEEN 41 AND 50 THEN 5
            ELSE NULL
        END AS year
    FROM
        episodes
),
CarbsPerEpisode AS (
    SELECT
        e.year,
        r.id_rec,
        r.carbs
    FROM
        EpisodeYears e
    JOIN
        chefs_episodes ce ON e.no_ep = ce.chef_no_ep
    JOIN
        recipes r ON ce.rec_id = r.id_rec
)
SELECT
    year,
    AVG(carbs) AS avg_carbs
FROM
    CarbsPerEpisode
GROUP BY
    year
ORDER BY
    year;
```

Αποτελέσματα Query:

```
 year |     avg_carbs
------+--------------------
    1 | 175.95043999999987
    2 | 171.16852999999992
    3 | 196.97043999999997
    4 |          160.28203
    5 | 189.91383999999994
(5 rows)
```

3.10. Ποιες Εθνικές κουζίνες έχουν τον ίδιο αριθμό συμμετοχών σε διαγωνισμούς, σε διάστημα δύο συνεχόμενων ετών, με τουλάχιστον 3 συμμετοχές ετησίως

```
WITH EpisodeYears AS (
    SELECT
        no_ep,
        CASE
            WHEN no_ep BETWEEN 1 AND 10 THEN 1
            WHEN no_ep BETWEEN 11 AND 20 THEN 2
            WHEN no_ep BETWEEN 21 AND 30 THEN 3
            WHEN no_ep BETWEEN 31 AND 40 THEN 4
            WHEN no_ep BETWEEN 41 AND 50 THEN 5
            ELSE NULL
        END AS year
    FROM
        episodes
),
CuisineAppearances AS (
    SELECT
        nc.name AS cuisine,
        ey.year,
        COUNT(*) AS appearances
    FROM
        national_cuisine nc
    JOIN
        chefs_episodes ce ON nc.id_national = ce.id_national
    JOIN
        EpisodeYears ey ON ce.chef_no_ep = ey.no_ep
    GROUP BY
        nc.name, ey.year
    HAVING
        COUNT(*) >= 3
),
CuisineAppearancesConsecutiveYears AS (
    SELECT
        c1.cuisine,
        c1.year AS year1,
        c2.year AS year2,
        c1.appearances AS appearances1,
        c2.appearances AS appearances2
    FROM
        CuisineAppearances c1
    JOIN
        CuisineAppearances c2 ON c1.cuisine = c2.cuisine AND c1.year = c2.year - 1
    WHERE
        c1.appearances = c2.appearances
)
SELECT
    cuisine,
    year1,
    year2,
    appearances1 AS appearances
FROM
    CuisineAppearancesConsecutiveYears
```

```
ORDER BY
    cuisine, year1;
```

Αποτελέσματα Query:

```
   cuisine     | year1 | year2 | appearances
---------------+-------+-------+-------------
 Arabia        |     2 |     3 |           5
 Arabia        |     4 |     5 |           6
 Argentina     |     3 |     4 |           3
 Canada        |     4 |     5 |           3
 France        |     1 |     2 |          10
 Italy         |     1 |     2 |           9
 Japan         |     2 |     3 |           6
 Kuwait        |     1 |     2 |           3
 Malaysian     |     4 |     5 |           3
 Peru          |     3 |     4 |           4
 United States |     1 |     2 |           9
 United States |     2 |     3 |           9
 United States |     3 |     4 |           9
 United States |     4 |     5 |           9
(14 rows)
```

3.11. Βρείτε τους top-5 κριτές που έχουν δώσει συνολικά την υψηλότερη βαθμολόγηση σε ένα μάγειρα. (όνομα κριτή, όνομα μάγειρα και συνολικό σκορ βαθμολόγησης)

```
WITH JudgeScores AS (
    SELECT
        s.judge_id,
        s.chef_id,
        SUM(s.score) AS total_score
    FROM
        score s
    GROUP BY
        s.judge_id, s.chef_id
),
JudgeChefNames AS (
    SELECT
        js.judge_id,
        js.chef_id,
        js.total_score,
        j.chef_name AS judge_name,
        c.chef_name AS chef_name
    FROM
        JudgeScores js
    JOIN
        chefs j ON js.judge_id = j.chef_id
    JOIN
        chefs c ON js.chef_id = c.chef_id
)
SELECT
    judge_name,
    chef_name,
    total_score
FROM
    JudgeChefNames
ORDER BY
    total_score DESC
LIMIT 5;
```

Αποτελέσματα Query:

```
 judge_name | chef_name | total_score
------------+-----------+-------------
 Nickolaus  | Samuele   |          13
 Ferdinande | Gannon    |          13
 Mervin     | Marjie    |          13
 Mae        | Howard    |          13
 Wren       | Haleigh   |          13
(5 rows)
```

3.12. Ποιο ήταν το πιο τεχνικά δύσκολο, από πλευράς συνταγών, επεισόδιο του διαγωνισμού ανά έτος;

```
WITH EpisodeYears AS (
    SELECT
        no_ep,
        CASE
            WHEN no_ep BETWEEN 1 AND 10 THEN 1
            WHEN no_ep BETWEEN 11 AND 20 THEN 2
            WHEN no_ep BETWEEN 21 AND 30 THEN 3
            WHEN no_ep BETWEEN 31 AND 40 THEN 4
            WHEN no_ep BETWEEN 41 AND 50 THEN 5
        END AS year
    FROM
        episodes
),
EpisodeDifficulty AS (
    SELECT
        ey.year,
        e.no_ep,
        AVG(d.id_diff) AS avg_difficulty
    FROM
        EpisodeYears ey
    JOIN
        chefs_episodes ce ON ey.no_ep = ce.chef_no_ep
    JOIN
        recipes r ON ce.rec_id = r.id_rec
    JOIN
        difficulty d ON r.id_diff = d.id_diff
    JOIN
        episodes e ON ey.no_ep = e.no_ep
    GROUP BY
        ey.year, e.no_ep
),
MaxDifficultyPerYear AS (
    SELECT
        year,
        MAX(avg_difficulty) AS max_difficulty
    FROM
        EpisodeDifficulty
    GROUP BY
        year
)
SELECT
    ed.year,
    ed.no_ep,
    ed.avg_difficulty
FROM
    EpisodeDifficulty ed
JOIN
    MaxDifficultyPerYear md ON ed.year = md.year AND ed.avg_difficulty = md.max_diff
ORDER BY
    ed.year;
```

Αποτελέσματα Query:

```
 year | no_ep |   avg_difficulty
------+-------+--------------------
    1 |     9 | 3.0000000000000000
    2 |    20 | 3.2000000000000000
    3 |    22 | 2.6000000000000000
    4 |    38 | 3.0000000000000000
    5 |    43 | 3.3000000000000000
(5 rows)
```

3.13. Ποιο επεισόδιο συγκέντρωσε τον χαμηλότερο βαθμό επαγγελματικής κατάρτισης (κριτές και μάγειρες);

```
WITH CharacterizeLevels AS (
    SELECT
        chef_id,
        CASE characterize
            WHEN 'C chef' THEN 1
            WHEN 'B chef' THEN 2
            WHEN 'A chef' THEN 3
            WHEN 'sous chef' THEN 4
            WHEN 'head chef' THEN 5
            ELSE 0
        END AS experience_level
    FROM
        chefs
),
ChefExperience AS (
    SELECT
        ce.chef_no_ep AS no_ep,
        AVG(cl.experience_level) AS avg_chef_experience
    FROM
        chefs_episodes ce
    JOIN
        CharacterizeLevels cl ON ce.chef_id = cl.chef_id
    GROUP BY
        ce.chef_no_ep
),
JudgeExperience AS (
    SELECT
        je.judge_no_ep AS no_ep,
        AVG(cl.experience_level) AS avg_judge_experience
    FROM
        judge_episodes je
    JOIN
        CharacterizeLevels cl ON je.judge_id = cl.chef_id
    GROUP BY
        je.judge_no_ep
),
EpisodeExperience AS (
    SELECT
        e.no_ep,
        COALESCE(ce.avg_chef_experience, 0) AS avg_chef_experience,
        COALESCE(je.avg_judge_experience, 0) AS avg_judge_experience,
        (COALESCE(ce.avg_chef_experience, 0) + COALESCE(je.avg_judge_experience, 0))      AS
    FROM
        episodes e
    LEFT JOIN
        ChefExperience ce ON e.no_ep = ce.no_ep
    LEFT JOIN
        JudgeExperience je ON e.no_ep = je.no_ep
)
SELECT
    ee.no_ep,
    ee.avg_total_experience
FROM
```

```
    EpisodeExperience ee
ORDER BY
    ee.avg_total_experience ASC
LIMIT 1;
```

Αποτελέσματα Query:

```
 no_ep |  avg_total_experience
-------+------------------------
    42 | 1.55000000000000000000
(1 row)
```

3.14. Ποια θεματική ενότητα έχει εμφανιστεί τις περισσότερες φορές στο διαγωνισμό;

```
SELECT
    tc.name AS theme_name,
    COUNT(rt.id_theme) AS appearance_count
FROM
    recipes_theme rt
JOIN
    theme_chapters tc ON rt.id_theme = tc.theme_chapters_id
GROUP BY
    tc.name
ORDER BY
    appearance_count DESC
LIMIT 1;
```

Αποτελέσματα Query:

```
 theme_name | appearance_count
------------+------------------
 Date Night |                8
(1 row)
```

3.15. Ποιες ομάδες τροφίμων δεν έχουν εμφανιστεί ποτέ στον διαγωνισμό;

```
WITH FoodGroupsInCompetition AS (
    SELECT DISTINCT
        fg.id,
        fg.name
    FROM
        food_groups fg
    JOIN
        recipes r ON fg.id = r.id_food_group
    JOIN
        chefs_episodes ce ON r.id_rec = ce.rec_id
)
SELECT
    fg.id,
    fg.name
FROM
    food_groups fg
LEFT JOIN
    FoodGroupsInCompetition fgc ON fg.id = fgc.id
WHERE
    fgc.id IS NULL;
```

Αποτελέσματα Query:

```
 id |              name
----+---------------------------------
  1 | Spices and Essential Oils
  2 | Coffee-Tea and Related Products
  3 | Preserved Foods
  4 | Sweeteners
  5 | Fats and Oils
 11 | Products with Sweeteners
 12 | Various Beverages
(7 rows)
```

10 DDL script

```
-- Create food_groups table
CREATE TABLE food_groups (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    description TEXT NOT NULL
);

-- Create difficulty table
CREATE TABLE difficulty (
    id_diff SERIAL PRIMARY KEY,
    description TEXT NOT NULL
);

-- Create national_cuisine table
CREATE TABLE national_cuisine (
    id_national SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL
);

-- Create recipes table
CREATE TABLE recipes (
    id_rec SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    description TEXT NOT NULL,
    tips TEXT NOT NULL,
    preparation_time INT NOT NULL CHECK(preparation_time > 0),
    execution_time INT NOT NULL CHECK(execution_time > 0),
    portions INT NOT NULL CHECK(portions > 0),
    fat FLOAT NOT NULL CHECK(fat >= 0),
    kcal FLOAT NOT NULL CHECK(kcal >= 0),
    protein FLOAT NOT NULL CHECK(protein >= 0),
    carbs FLOAT NOT NULL CHECK(carbs >= 0),
    id_food_group INTEGER REFERENCES food_groups(id),
    id_diff INTEGER REFERENCES difficulty(id_diff),
    id_national INTEGER REFERENCES national_cuisine(id_national)
);

-- Create meal_type table
CREATE TABLE meal_type (
        id_meal_type INTEGER primary KEY,
    type_name VARCHAR(50) NOT NULL
);

-- Create meal_type_rec table
CREATE TABLE meal_type_rec (
    id_rec INTEGER REFERENCES recipes(id_rec) ON DELETE RESTRICT ON UPDATE CASCADE,
    id_type INTEGER REFERENCES meal_type(id_meal_type) ON DELETE RESTRICT,
    PRIMARY KEY (id_rec, id_type)
);

-- Create tags table
CREATE TABLE tags (
    id_tags SERIAL PRIMARY key,
    name VARCHAR(50)
```

```
);

-- Create tags_rec table
CREATE TABLE tags_rec (
    id_tags INTEGER REFERENCES tags(id_tags),
    id_rec INTEGER REFERENCES recipes(id_rec),
    PRIMARY KEY (id_tags, id_rec)
);

-- Create ingredients table
CREATE TABLE ingredients (
    ing_id SERIAL PRIMARY KEY,
    ing_name VARCHAR(50) NOT NULL,
    kcal FLOAT NOT NULL CHECK(kcal >= 0),
    fat FLOAT NOT NULL CHECK(fat >= 0),
    protein FLOAT NOT NULL CHECK(protein >= 0),
    carbs FLOAT NOT NULL CHECK(carbs >= 0),
    food_groups_id INTEGER REFERENCES food_groups(id)
);

-- Create ingredient_recipes table
CREATE TABLE ingredient_recipes (
    id_ing INTEGER REFERENCES ingredients(ing_id) ON DELETE RESTRICT ON UPDATE CASCADE
    id_rec INTEGER REFERENCES recipes(id_rec) ON DELETE RESTRICT ON UPDATE CASCADE,
    quantity TEXT NOT NULL,
    basic VARCHAR(50) NOT NULL,
    PRIMARY KEY (id_ing, id_rec)
);

-- Create equipment table
CREATE TABLE equipment (
    eq_id SERIAL PRIMARY KEY,
    eq_name VARCHAR(50) NOT NULL,
    instructions TEXT NOT NULL
);

CREATE TABLE recipes_has_equipment (
    recipes_id INTEGER REFERENCES recipes(id_rec),
    equipment_id INTEGER REFERENCES equipment(eq_id),
    PRIMARY KEY (recipes_id, equipment_id)
);


-- Create steps table
CREATE TABLE steps (
    steps_id SERIAL PRIMARY KEY,
    description TEXT NOT null,
    recipe_id INTEGER references recipes(id_rec),
    order_no INTEGER
);




-- Create theme_chapters table
CREATE TABLE theme_chapters (
```

```
    theme_chapters_id SERIAL PRIMARY KEY,
    name VARCHAR(50),
    description TEXT
);

-- Create recipes_theme table
CREATE TABLE recipes_theme (
    id_rec INTEGER REFERENCES recipes(id_rec) ON DELETE RESTRICT ON UPDATE CASCADE,
    id_theme INTEGER REFERENCES theme_chapters(theme_chapters_id) ON DELETE RESTRICT
    PRIMARY KEY (id_rec, id_theme)
);

-- Create episodes table
CREATE TABLE episodes (
    no_ep SERIAL PRIMARY KEY
);

-- Create chefs table
CREATE TABLE chefs (
    chef_id SERIAL PRIMARY KEY,
    chef_name VARCHAR(50) NOT NULL,
    surname VARCHAR(50) NOT NULL,
    phone VARCHAR(10) NOT NULL CHECK(LENGTH(phone) = 10),
    birth DATE NOT NULL,
    chef_age INT NOT NULL CHECK(chef_age > 0),
    experience INT NOT NULL CHECK(experience > 0),
    characterize TEXT NOT NULL,
    pas VARCHAR(50) not null
    CHECK(chef_age = (EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM birth))
        OR chef_age = (EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM birth)) +
);


-- Create chefs_episodes table
CREATE TABLE chefs_episodes (
    chef_id INTEGER REFERENCES chefs(chef_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    chef_no_ep INTEGER REFERENCES episodes(no_ep) ON DELETE RESTRICT ON UPDATE CASCADE
    id_national INTEGER REFERENCES national_cuisine(id_national) ON DELETE RESTRICT,
    rec_id INTEGER REFERENCES recipes(id_rec) ON DELETE RESTRICT ON UPDATE CASCADE,
    PRIMARY KEY (chef_id, chef_no_ep)
);

-- Create judge_episodes table
CREATE TABLE judge_episodes (
    judge_id INTEGER REFERENCES chefs(chef_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    judge_no_ep INTEGER REFERENCES episodes(no_ep) ON DELETE RESTRICT ON UPDATE CASCADE
    PRIMARY KEY (judge_id, judge_no_ep)
);

-- Create score table
CREATE TABLE score (
    chef_id INTEGER,
    chef_no_ep INTEGER,
    judge_id INTEGER,
    judge_no_ep INTEGER,
    score INTEGER,
```

```
    PRIMARY KEY (chef_id, chef_no_ep, judge_id, judge_no_ep),
    FOREIGN KEY (chef_id, chef_no_ep) REFERENCES chefs_episodes(chef_id, chef_no_ep) ON DELET
    FOREIGN KEY (judge_id, judge_no_ep) REFERENCES judge_episodes(judge_id, judge_no      ON D
);

-- Create chef_national table
CREATE TABLE chef_national (
    id_chef INTEGER REFERENCES chefs(chef_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    id_national INTEGER REFERENCES national_cuisine(id_national) ON DELETE RESTRICT
    PRIMARY KEY (id_chef, id_national)
);



CREATE OR REPLACE FUNCTION convert_quantity_to_factor(quantity_text TEXT) RETURNS FLOAT AS
DECLARE
    numeric_part FLOAT;
    unit_part TEXT;
BEGIN
    -- Extract numeric part from the quantity text
    numeric_part := substring(quantity_text from '^[0-9]+')::FLOAT;

    -- Determine the unit (assuming only 'g' or 'ml' are used)
    unit_part := substring(quantity_text from '[a-zA-Z]+$');

    -- Convert based on unit
    IF unit_part = 'g' THEN
        RETURN numeric_part / 100.0;
    ELSIF unit_part = 'ml' THEN
        RETURN numeric_part / 100.0;
    ELSE
        -- Default case to handle unexpected unit
        RETURN NULL;
    END IF;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION recipes_nutrition_insert_trigger() RETURNS TRIGGER AS $$
DECLARE
    total_kcal FLOAT := 0;
    total_fat FLOAT := 0;
    total_protein FLOAT := 0;
    total_carbs FLOAT := 0;
    quantity_factor FLOAT;
    rec RECORD;
BEGIN
    -- Calculate total nutritional values
    FOR rec IN (
        SELECT i.ing_id, i.kcal, i.fat, i.protein, i.carbs, ir.quantity
        FROM ingredients i
        JOIN ingredient_recipes ir ON i.ing_id = ir.id_ing
        WHERE ir.id_rec = NEW.id_rec
    ) LOOP
        quantity_factor := convert_quantity_to_factor(rec.quantity);
```

```
        -- Debugging output
        RAISE NOTICE 'Ingredient: %, Quantity: %, Factor: %', rec.ing_id, rec.quanti

        -- Ensure the quantity factor is not null
        IF quantity_factor IS NOT NULL THEN
            total_kcal := total_kcal + (rec.kcal * quantity_factor);
            total_fat := total_fat + (rec.fat * quantity_factor);
            total_protein := total_protein + (rec.protein * quantity_factor);
            total_carbs := total_carbs + (rec.carbs * quantity_factor);
        ELSE
            RAISE NOTICE 'Skipping ingredient % due to null quantity factor', rec.ir
        END IF;
    END LOOP;

    -- Update the recipe row with calculated totals
    UPDATE recipes
    SET kcal = total_kcal,
        fat = total_fat,
        protein = total_protein,
        carbs = total_carbs
    WHERE id_rec = NEW.id_rec;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE role admin with password 'your_admin_password';

GRANT ALL PRIVILEGES ON DATABASE test TO admin;



CREATE ROLE chef_user;


CREATE OR REPLACE FUNCTION create_chef_user()
RETURNS TRIGGER AS $$
BEGIN

    -- Create a new role with the chef_id as both the username and password

    EXECUTE format('CREATE ROLE %I WITH LOGIN PASSWORD %L', NEW.chef_id, NEW.pas);

    -- Grant INSERT and UPDATE privileges on the recipes table to the new role

    EXECUTE format('GRANT INSERT, UPDATE ON TABLE recipes TO %I', NEW.chef_id);



    -- Grant INSERT and UPDATE privileges on the chefs table to the new role

    EXECUTE format('GRANT INSERT, UPDATE ON TABLE chefs TO %I', NEW.chef_id);



    RETURN NEW;
```

```
END;

$$ LANGUAGE plpgsql;



CREATE TRIGGER after_chef_insert
AFTER INSERT ON chefs
FOR EACH ROW
EXECUTE FUNCTION create_chef_user();

-- Drop existing trigger if it exists
DROP TRIGGER IF EXISTS update_recipe_nutrition ON ingredient_recipes;

-- Create the trigger to call the function after insert or update
CREATE TRIGGER update_recipe_nutrition
AFTER INSERT OR UPDATE ON ingredient_recipes
FOR EACH ROW
EXECUTE FUNCTION recipes_nutrition_insert_trigger();



--views
-- Create view for each recipe and its ingredients in a list
CREATE VIEW recipe_ingredients AS
SELECT
    r.name AS recipe_name,
    array_agg(i.ing_name) AS ingredients
FROM
    recipes r
JOIN
    ingredient_recipes ir ON r.id_rec = ir.id_rec
JOIN
    ingredients i ON ir.id_ing = i.ing_id
GROUP BY
    r.name;


-- Create view for the total points per season per chef
CREATE VIEW chef_total_points_per_season AS
WITH season_episodes AS (
    SELECT 1 AS season, 1 AS start_ep, 10 AS end_ep UNION ALL
    SELECT 2 AS season, 11 AS start_ep, 20 AS end_ep UNION ALL
    SELECT 3 AS season, 21 AS start_ep, 30 AS end_ep UNION ALL
    SELECT 4 AS season, 31 AS start_ep, 40 AS end_ep UNION ALL
    SELECT 5 AS season, 41 AS start_ep, 50 AS end_ep
),
season_scores AS (
    SELECT
        s.chef_id,
        se.season,
        SUM(s.score) AS total_score
    FROM
        score s
    JOIN
```

```
        chefs_episodes ce ON s.chef_id = ce.chef_id AND s.chef_no_ep = ce.chef_no_ep
    JOIN
        season_episodes se ON s.chef_no_ep BETWEEN se.start_ep AND se.end_ep
    GROUP BY
        s.chef_id, se.season
)
SELECT
    c.chef_id,
    c.chef_name,
    c.surname,
    ss.season,
    ss.total_score
FROM
    chefs c
JOIN
    season_scores ss ON c.chef_id = ss.chef_id
order by season,total_score;

--Indexes
--tags_rec index
CREATE INDEX idx_tags_rec_id_rec ON tags_rec ( id_rec );
CREATE INDEX idx_tags_rec_id_tags ON tags_rec ( id_tags );
--chef_episode index
CREATE INDEX idx_chefs_episodes_no_ep ON chefs_episodes ( chef_no_ep );
--recipes_equipment index
CREATE INDEX idx_recipes_has_equipment_recipes_id ON recipes_has_equipment ( recipes
--ingredient index to search fast by name
CREATE INDEX idx_ingredients_name ON ingredients (ing_name);
--index in score because there are 1500 tuples
CREATE INDEX idx_score_chef_id ON score (chef_id);
CREATE INDEX idx_score_chef_no_ep ON score (chef_no_ep);
CREATE INDEX idx_score_judge_id ON score (judge_id);
CREATE INDEX idx_score_judge_no_ep ON score (judge_no_ep);
```