Εθνικό Μετσόβιο Πολυτεχνείο

Βάσεις Δεδομένων
Αναφορά Εξαμηνιαίας Εργασίας

Ομάδα 15

Αντωνία Πάπαλου - ge20706 - papalouantonia@gmail.com

Σωτήρης Καλλίσιης - ge20723 - kallishis.sotiris@gmail.com

Github Repository: https://github.com/kallishis/Databases.git

Μάιος 2024

Περιεχόμενα

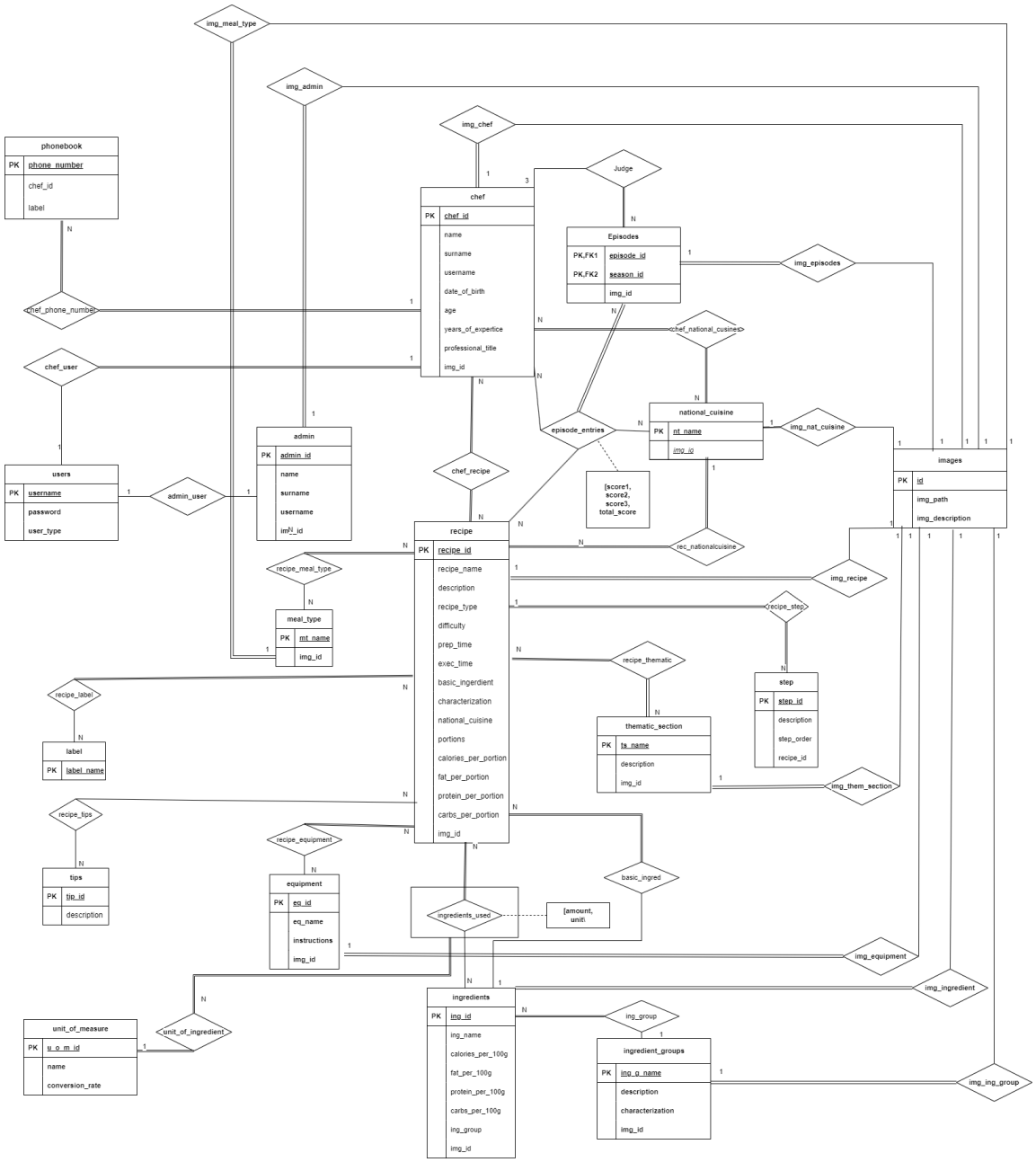
Εισαγωγή

Στα πλαίσια του μαθήματος Βάσεις Δεδομένων κληθήκαμε να δημιουργήσουμε μια βάση δεδομένων για ένα διαγωνισμό μαγειρικής στην οποία αποθηκεύονται πληροφορίες για συντεγές, μάγειρες, υλικά, επεισόδια κ.α.

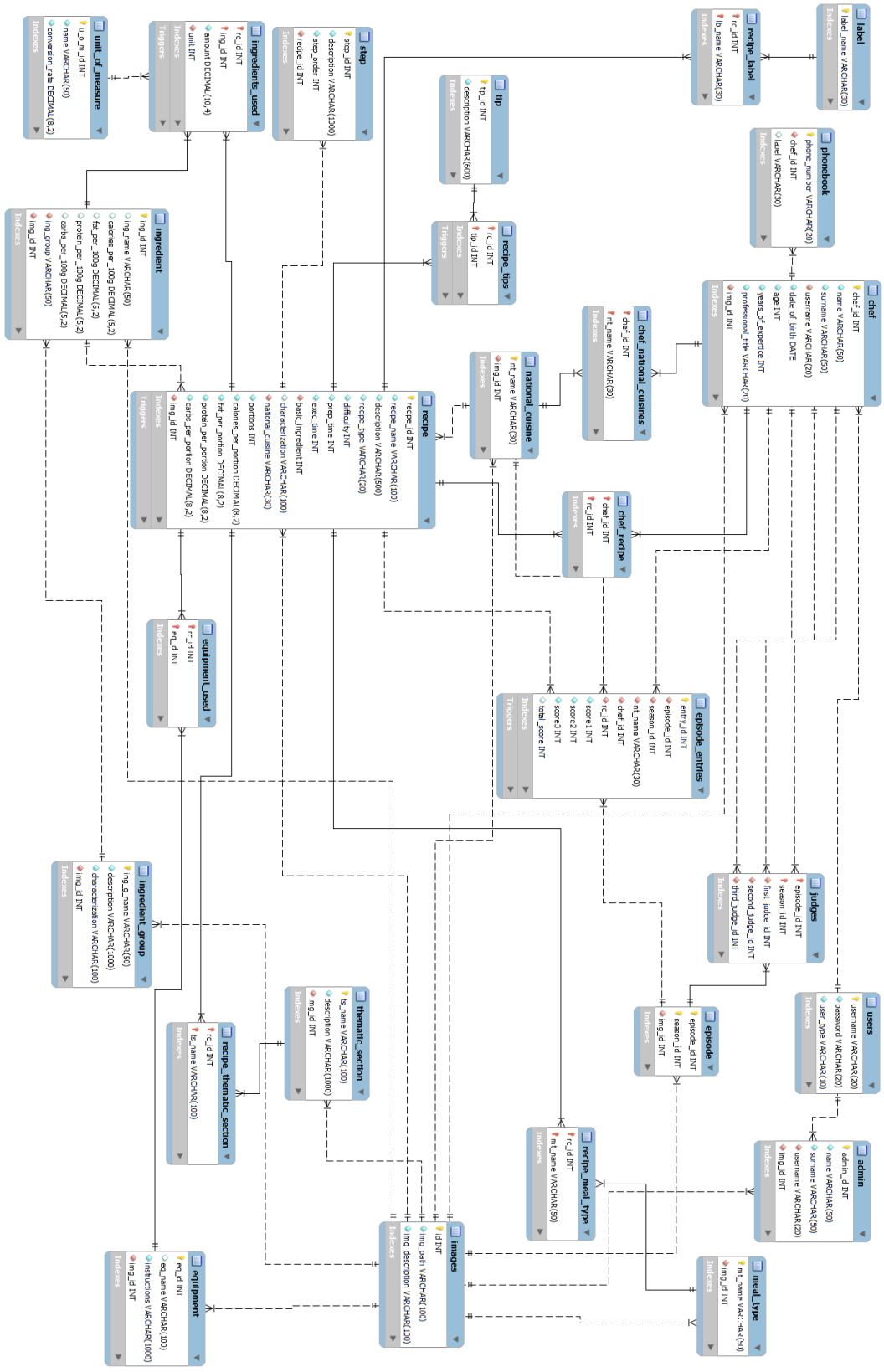
Η αναφορά αυτή περιέχει τη διαδικασία ανάπτυξης της εν λόγω βάσης, τους κώδικες για την κατασκευή της βάσης και την εισαγωγή δεδομένων σε αυτή, τους κώδικες για την κλήρωση τυχαίων επεισοδίων βάση κάποιων περιορισμών καθώς και κάποια ερωτήματα(queries) που κληθήκαμε να απαντήσουμε σε SQL.

Η ανάπτυξη της βάσης έγινε σε MySQL ενώ η κλήρωση των τυχαίων επεισοδίων έγινε σε Node JS, όπου χρησιμοποιήθηκε το Visual Studio Code.

1. Διάγραμμα ER (Entity-Relation Diagram)



2. Σχεσιακό Διάγραμμα (Schema Diagram)



3. Περιορισμοί (Constraints) Βάσης Δεδομένων

Οι περιορισμοί της βάσης δεδομένων μπορούν να χωριστούν στις εξής 5 κατηγορίες και παρουσιάζουμε παραδείγματα από τους περιορισμούς που έχουμε για τη βάση μας σε κάθε κατηγορία. Αναλυτικά οι περιορισμοί που έχουν οριστεί για τη βάση δεδομένων φαίνονται στο ddl script. :

- Primary Key Constraints: Διασφαλίζει μοναδικότητα εγγραφών,δεν επιτρέπει τιμές NULL.
  - Στον πίνακα recipe έχουμε πρωτεύον κλειδί τη στήλη recipe_id
  - Στον πίνακα label έχουμε πρωτεύον κλειδί τη στήλη label_name
  - Στον πίνακα recipe_label έχουμε πρωτεύον κλειδί τον συνδυασμό των στηλών rc_id και lb_name
  - Στον πίνακα phonebook έχουμε πρωτεύον κλειδί τη στήλη phone_number
  - Στον πίνακα national_cuisine έχουμε πρωτεύον κλειδί τη στήλη nt_name
  - Στον πίνακα tip έχουμε πρωτεύον κλειδί τη στήλη tip_id
  - Στον πίνακα chef έχουμε πρωτεύον κλειδί τη στήλη chef_id
  - Στον πίνακα chef_national_cuisines έχουμε πρωτεύον κλειδί τον συνδυασμό των στηλών chef_id και nt_name
  - Στον πίνακα chef_recipe έχουμε πρωτεύον κλειδί τον συνδυασμό των στηλών chef_id και rc_id
  - Στον πίνακα recipe_tips έχουμε πρωτεύον κλειδί τον συνδυασμό των στηλών rc_id και tip_id
  - Στον πίνακα step έχουμε πρωτεύον κλειδί τη στήλη step_id
  - Στον πίνακα ingredients_used έχουμε πρωτεύον κλειδί τον συνδυασμό των στηλών rc_id και ing_id
  - Στον πίνακα unit_of_measure έχουμε πρωτεύον κλειδί τη στήλη u_o_m_id
  - Στον πίνακα ingredient έχουμε πρωτεύον κλειδί τη στήλη ing_id
  - Στον πίνακα equipment έχουμε πρωτεύον κλειδί τη στήλη eq_id
  - Στον πίνακα equipment_used έχουμε πρωτεύον κλειδί τον συνδυασμό των στηλών rc_id και eq_id
  - Στον πίνακα episode έχουμε πρωτεύον κλειδί τον συνδυασμό των στηλών episode_id και season_id
  - Στον πίνακα episode_entries έχουμε πρωτεύον κλειδί τη στήλη entry_id
  - Στον πίνακα judges έχουμε πρωτεύον κλειδί τον συνδυασμό των στηλών episode_id και season_id
  - Στον πίνακα users έχουμε πρωτεύον κλειδί τη στήλη username
  - Στον πίνακα admin έχουμε πρωτεύον κλειδί τη στήλη admin_id
  - Στον πίνακα meal_type έχουμε πρωτεύον κλειδί τη στήλη mt_name
  - Στον πίνακα recipe_meal_type έχουμε πρωτεύον κλειδί τον συνδυασμό των στηλών rc_id και mt_name

  - Στον πίνακα thematic_section έχουμε πρωτεύον κλειδί τη στήλη ts_name
  - Στον πίνακα recipe_thematic_section έχουμε πρωτεύον κλειδί τον συνδυασμό των στηλών rc_id και ts_name
  - Στον πίνακα images έχουμε πρωτεύον κλειδί τη στήλη id
  - Στον πίνακα ingredient_group έχουμε πρωτεύον κλειδί τη στήλη ing_g_name

- Foreign Key Constraints: Συνδέει εγγραφές μεταξύ πινάκων, εξασφαλίζει ακεραιότητα δεδομένων.
  Σε όλα τα δευτερεύοντα κλειδιά έχει μπει ο περιορισμός ON DELETE RESTRICT ON UPDATE CASCADE έτσι ώστε να μην επιτρέπεται η διαγραφή κάποιας εγγραφής που είναι δευτερεύον κλειδί σε άλλο πίνακα και αν ενημερώνεται, τότε να ενημερώνονται και οι εγγραφές των αντίστοιχων πινάκων με τους οποίους συνδέεται.
  - Στον πίνακα meal_type έχουμε δευτερεύον κλειδί τη στήλη img_id που αναφέρεται στη στήλη id του πίνακα images
  - Στον πίνακα equipment έχουμε δευτερεύον κλειδί τη στήλη img_id που αναφέρεται στη στήλη id του πίνακα images
  - Στον πίνακα ingredient έχουμε δευτερεύον κλειδί τη στήλη img_id που αναφέρεται στη στήλη id του πίνακα images
  - Στον πίνακα ingredient έχουμε δευτερεύον κλειδί τη στήλη ing_group που αναφέρεται στη στήλη ing_g_name του πίνακα ingredient_group
  - Στον πίνακα thematic_section έχουμε δευτερεύον κλειδί τη στήλη img_id που αναφέρεται στη στήλη id του πίνακα images
  - Στον πίνακα chef έχουμε δευτερεύον κλειδί τη στήλη img_id που αναφέρεται στη στήλη id του πίνακα images
  - Στον πίνακα chef έχουμε δευτερεύον κλειδί τη στήλη username που αναφέρεται στη στήλη username του πίνακα users
  - Στον πίνακα phonebook έχουμε δευτερεύον κλειδί τη στήλη chef_id που αναφέρεται στη στήλη chef_id του πίνακα chef
  - Στον πίνακα recipe έχουμε δευτερεύον κλειδί τη στήλη img_id που αναφέρεται στη στήλη id του πίνακα images
  - Στον πίνακα recipe έχουμε δευτερεύον κλειδί τη στήλη national_cuisine που αναφέρεται στη στήλη nt_name του πίνακα national_cuisine
  - Στον πίνακα recipe έχουμε δευτερεύον κλειδί τη στήλη basic_ingredient που αναφέρεται στη στήλη ing_id του πίνακα ingredient
  - Στον πίνακα step έχουμε δευτερεύον κλειδί τη στήλη recipe_id που αναφέρεται στη στήλη recipe_id του πίνακα recipe
  - Στον πίνακα recipe_meal_type έχουμε δευτερεύον κλειδί τη στήλη rc_id που αναφέρεται στη στήλη recipe_id του πίνακα recipe
  - Στον πίνακα recipe_meal_type έχουμε δευτερεύον κλειδί τη στήλη mt_name που αναφέρεται στη στήλη mt_name του πίνακα meal_type

- Στον πίνακα recipe_label έχουμε δευτερεύον κλειδί τη στήλη rc_id που αναφέρεται στη στήλη recipe_id του πίνακα recipe
- Στον πίνακα recipe_label έχουμε δευτερεύον κλειδί τη στήλη lb_name που αναφέρεται στη στήλη label_name του πίνακα label
- Στον πίνακα recipe_tips έχουμε δευτερεύον κλειδί τη στήλη rc_id που αναφέρεται στη στήλη recipe_id του πίνακα recipe
- Στον πίνακα recipe_tips έχουμε δευτερεύον κλειδί τη στήλη tip_id που αναφέρεται στη στήλη tip_id του πίνακα tip
- Στον πίνακα equipment_used έχουμε δευτερεύον κλειδί τη στήλη rc_id που αναφέρεται στη στήλη recipe_id του πίνακα recipe
- Στον πίνακα equipment_used έχουμε δευτερεύον κλειδί τη στήλη eq_id που αναφέρεται στη στήλη eq_id του πίνακα equipment
- Στον πίνακα ingredients_used έχουμε δευτερεύον κλειδί τη στήλη rc_id που αναφέρεται στη στήλη recipe_id του πίνακα recipe
- Στον πίνακα ingredients_used έχουμε δευτερεύον κλειδί τη στήλη ing_id που αναφέρεται στη στήλη ing_id του πίνακα ingredient
- Στον πίνακα ingredients_used έχουμε δευτερεύον κλειδί τη στήλη unit που αναφέρεται στη στήλη u_o_m_id του πίνακα unit_of_measure
- Στον πίνακα recipe_thematic_section έχουμε δευτερεύον κλειδί τη στήλη rc_id που αναφέρεται στη στήλη recipe_id του πίνακα recipe
- Στον πίνακα recipe_thematic_section έχουμε δευτερεύον κλειδί τη στήλη ts_name που αναφέρεται στη στήλη ts_name του πίνακα thematic_section
- Στον πίνακα chef_national_cuisines έχουμε δευτερεύον κλειδί τη στήλη national_cuisine που αναφέρεται στη στήλη nt_name του πίνακα national_cuisine
- Στον πίνακα chef_national_cuisines έχουμε δευτερεύον κλειδί τη στήλη chef_id που αναφέρεται στη στήλη chef_id του πίνακα chef
- Στον πίνακα chef_recipe έχουμε δευτερεύον κλειδί τη στήλη rc_id που αναφέρεται στη στήλη recipe_id του πίνακα recipe
- Στον πίνακα chef_recipe έχουμε δευτερεύον κλειδί τη στήλη chef_id που αναφέρεται στη στήλη chef_id του πίνακα chef
- Στον πίνακα episode έχουμε δευτερεύον κλειδί τη στήλη img_id που αναφέρεται στη στήλη id του πίνακα images
- Στον πίνακα episode_entries έχουμε δευτερεύον κλειδί τον συνδυασμό των στηλών episode_id και season_id που αναφέρεται στις στήλες episode_id και season_id του πίνακα episode
- Στον πίνακα episode_entries έχουμε δευτερεύον κλειδί τη στήλη rc_id που αναφέρεται στη στήλη recipe_id του πίνακα recipe
- Στον πίνακα episode_entries έχουμε δευτερεύον κλειδί τη στήλη chef_id που αναφέρεται στη στήλη chef_id του πίνακα chef
- Στον πίνακα episode_entries έχουμε δευτερεύον κλειδί τη στήλη nt_name που αναφέρεται στη στήλη nt_name του πίνακα national_cuisine

  - Στον πίνακα judges έχουμε δευτερεύον κλειδί τον συνδυασμό των στηλών episode_id και season_id που αναφέρεται στις στήλες episode_id και season_id του πίνακα episode
  - Στον πίνακα judges έχουμε δευτερεύον κλειδί τη στήλη first_judge_id που αναφέρεται στη στήλη chef_id του πίνακα chef
  - Στον πίνακα judges έχουμε δευτερεύον κλειδί τη στήλη second_judge_id που αναφέρεται στη στήλη chef_id του πίνακα chef
  - Στον πίνακα judges έχουμε δευτερεύον κλειδί τη στήλη third_judge_id που αναφέρεται στη στήλη chef_id του πίνακα chef

- Unique Constraints : Επιβάλλει μοναδικότητα τιμών σε στήλη ή στήλες, επιτρέπει μία τιμή NULL.
  - Στο πίνακα tip η στήλη description έχει τον περιορισμό Unique αφού δεν επιτρέπουμε σε δύο διαφορετικά tip να έχουν την ίδια περιγραφή.

- Not null Constraints : Απαγορεύει τις NULL τιμές σε μια στήλη.
  Σχεδόν όλες οι στήλες παίρνουν αυτό τον περιορισμό ώστε να μην υπάρχουν κενά στα δεδομένα μας.

- Check Constraints : Επιβάλλει συγκεκριμένες συνθήκες στις τιμές μιας στήλης.
  - Στο πίνακα ingredient υπάρχουν περιορισμοί Check στις στήλες calories_per_100g,fat_per_100g, protein_per_100g, carbs_per_100g που διασφαλίζουν την ορθότητα των τιμών των στηλών. Επίσης υπάρχει ακόμα ένας check περιορισμός με όνομα sum_of_macros το οποίο διασφαλίζει ότι το άθροισμα των macros ενός συστατικού δεν ξεπερνά τα 100 γραμμάρια.
  - Στον πίνακα users υπάρχει ένας check περιορισμός με όνομα password_length που ελέγχει αν τα δεδομένα της στήλη password έχουν το σωστό μήκος και ένας check περιορισμός με όνομα user_type_accepted_values που ελέγχει αν οι τιμές της στήλης user_type είναι οι επιτρεπτές.
  - Στον πίνακα users υπάρχει ένας check περιορισμός στις στήλες age και year_of_expertice που διασφαλίζει ότι είναι θετικοί αριθμοί.
  - Στον πίνακα unit_of_measure υπάρχει ένας check περιορισμός στη στήλη conversion_rate που διασφαλίζει ότι είναι θετικός αριθμός.
  - Στον πίνακα recipe υπάρχουν check περιορισμοί στις στήλες difficulty, prep_time, exec_time, portions, calories_per_portion,fat_per_portion, protein_per_portion, carbs_per_portion που διασφαλίζουν ότι οι στήλες παίρνουν έγκυρες τιμές. Επίσης υπάρχει ακόμα ένας check περιορισμός με όνομα recipe_type_values το οποίο διασφαλίζει ότι το η στήλη recipe_type παίρνει επιτρεπτές τιμές.
  - Στον πίνακα ingredients_used υπάρχει ένας check περιορισμός στη στήλη amount που διασφαλίζει ότι είναι θετικός αριθμός.
  - Στον πίνακα episode_entries υπάρχουν check περιορισμοί στις στήλες score1,score2,score3 που διασφαλίζουν ότι παίρνουν τιμές από 1 μέχρι 5.

- Default : Ορίζει μια προεπιλεγμένη τιμή για μια στήλη αν δεν δοθεί άλλη τιμή.

– Στο πίνακα recipe οι στήλες portions, calories_per_portion,fat_per_portion, protein_per_portion, carbs_per_portion έχουν αρχική τιμή 0.

4. Triggers

Τα Triggers εκτελούν κάποιες διαδικασίες όταν συμβεί ένα συμβάν στη βάση, είτε αυτό είναι εισαγωγή δεδομένων, διαγραφή δεδομένων κλπ. Στη βάση μας εμείς έχουμε εισάγει κάποια Triggers, έτσι ώστε να διασφαλίσουμε κάποιους επιπλέον περιορισμούς που έχουμε για τα δεδομένα μας και να ανανεώσουμε τις τιμές κάποιων στηλών κάποιων πινάκων μετά την εισαγωγή συγκεκριμένων δεδομένων. Συκεκριμένα τα Triggers που εισάγαμε είναι τα εξής:

- update_nutritional_info_after_insert_t
  Ενημερώνει τα macros μιας συνταγής μετά από εισαγωγή μιας εγγραφής στον πίνακα ingredients_used.
- update_nutritional_info_after_delete_t
  Ενημερώνει τα macros μιας συνταγής μετά από δαιγραφή μιας εγγραφής στον πίνακα ingredients_used.
- update_nutritional_info_after_update_t
  Ενημερώνει τα macros μιας συνταγής μετά από ενημέρωση μιας εγγραφής στον πίνακα ingredients_used.
- autocharacterization
  Ενημερώνει την τιμή της στήλης characterization του πίνακα recipe μετά από εισαγωγή μιας εγγραφής στον συγκεκριμένο πίνακα με βάση την ομάδα τροφίμων του βασικού συστατικού της συνταγής.
- invalid_years_of_expertice
  Επιστρέφει μήνυμα λάθους αν ο χρήστης προσπαθήσει να εισάγει ένα chef με χρόνια υπηρεσίας περισσότερα από την ηλικία του.
- consecutive_episodes_check
  Επιστρέφει μήνυμα λάθους όταν πάμε να εισάγουμε μια εγγραφή στον πίνακα episode_entries για την οποία είτε ο chef, είτε η εθνική κουζίνα είτε η συνταγή συμμετέχει σε 4ο συνεχόμενο επεισόδιο.
- consecutive_judges_check
  Επιστρέφει μήνυμα λάθους όταν πάμε να εισάγουμε μια εγγραφή στον πίνακα judges για την οποία κάποιος κριτής συμμετέχει σε 4ο συνεχόμενο επεισόδιο.
- check_tip_recipe_limit
  Επιστρέφει μήνυμα λάθους όταν πάμε να προσθέσουμε μια εγγραφή στον πίνακα recipe_tips, όπου η συγκεκριμένη συνταγή έχει ήδη το μέγιστο επιτεπόμενο αριθμό tips (που είναι 3).

5. Stored Procedures

Τα stored procedures είναι συναρτήσεις που είναι αποθηκευμένες στη βάση και μπορούν να χρησιμοποιηθούν για διάφορες λειτουργίες. Στη βάση μας εισάγαμε τις εξής stored procedures:

- update_recipe_nutritional_info_after_delete
  Είναι μια συνάρτηση που ενημερώνει τα macros μια συνάρτησης μετά την προσθήκη κάποιου νέου υλικού, προσθέτωντας τα macros του υλικού επί την ποσότητα που χρησιμοποιείται στην συνταγή.
- update_recipe_nutritional_info_after_insert
  Είναι μια συνάρτηση που ενημερώνει τα macros μια συνάρτησης μετά την διαγραφή κάποιου υλικού, αφαιρώντας τα macros του υλικού επί την ποσότητα που χρησιμοποιούσαμε πριν τη διαγραφή στην συνταγή.
- GetChefsByCuisineAndSeason
  Είναι μια συνάρτηση που χρησιμοποιείται στο ερώτημα 3.2, όπου και θα την εξηγήσουμε περαιτέρω.

Έχουμε επίσης κάποια ακόμα stored procedures που δεν παρουσιάζονται στο ddl script αλλά στο roles.sql script τα οποία θα επεξηγήσουμε στην αντίστοιχη παράγραφο.

6. Indexes

Τα indexes (ευρετήρια) χρησιμοποιούνται από το rdbms για ταχύτερη εύρεση εγγραφών, οπότε είναι χρήσιμα για την ταχύτερη λειτουργία της βάσης δεδομένων σε διάφορα queries.
Η MySQL εισάγει αυτόματα indexes για όλα τα πρωτεύοντα και δευτερεύοντα κλειδιά των πινάκων, επομένω οι πλήστες από τις στήλες που χρησιμοποιούνται στα ερωτήματα είχαν ήδη ευρετήριο και δεν χρειάστηκε να δημιουργήσουμε εμείς.
Η μόνη στήλη στην οποία δημιουργήσαμε index είναι η στήλη age του πίνακα chef έτσι ώστε να τρέχει ταχύτερα το ερώτημα 3.3

7. DDL script

Το DDL(Data Definition Language) script αφορά την δημιουργία του σχήματος της βάσης, δηλαδή των πινάκων, των σχέσεων μεταξύ τους, των περιορισμών, των ευρετηρίων, των trigers, των stored procedure κτλ. Παρατίθεται παρακάτω το DDL script για την δική μας βάση cooking_competition:

```sql
DROP DATABASE IF EXISTS cooking_competition;

CREATE DATABASE cooking_competition;
USE cooking_competition;

/*--------------------------------------------------------TABLES--------------------------------------------------------*/
CREATE TABLE images (
        id INT AUTO_INCREMENT PRIMARY KEY,
    img_path VARCHAR(100) NOT NULL,
    img_description VARCHAR(100) NOT NULL
```

```
);
CREATE TABLE meal_type (
        mt_name VARCHAR(50) PRIMARY KEY,
    img_id INT NOT NULL,
        FOREIGN KEY (img_id) REFERENCES images(id) ON DELETE RESTRICT ON UPDATE CASCADE
);
CREATE TABLE label (
        label_name VARCHAR(30) PRIMARY KEY
);
CREATE TABLE tip (
        tip_id INT AUTO_INCREMENT PRIMARY KEY,
    description VARCHAR(600) UNIQUE NOT NULL
);
CREATE TABLE equipment(
        eq_id INT AUTO_INCREMENT PRIMARY KEY,
        eq_name VARCHAR(100) NOT NULL,
    instructions VARCHAR(1000) NOT NULL,
    img_id INT NOT NULL,
        FOREIGN KEY (img_id) REFERENCES images(id) ON DELETE RESTRICT ON UPDATE CASCADE
);
CREATE TABLE ingredient_group(
        ing_g_name VARCHAR(50) PRIMARY KEY,
    description VARCHAR(1000) NOT NULL,
    characterization VARCHAR(100) NOT NULL,
    img_id INT NOT NULL,
        FOREIGN KEY (img_id) REFERENCES images(id) ON DELETE RESTRICT ON UPDATE CASCADE
);
CREATE TABLE ingredient(
        ing_id INT AUTO_INCREMENT PRIMARY KEY,
        ing_name VARCHAR(50) NOT NULL,
    calories_per_100g NUMERIC(5,2) CHECK(calories_per_100g >= 0 AND calories_per_100g<=900),
    fat_per_100g NUMERIC(5,2) CHECK(fat_per_100g >= 0 AND fat_per_100g <= 100),
    protein_per_100g NUMERIC(5,2) CHECK(protein_per_100g >= 0 AND protein_per_100g <= 100),
    carbs_per_100g NUMERIC(5,2) CHECK(carbs_per_100g >= 0 AND carbs_per_100g <= 100),
    ing_group VARCHAR(50) NOT NULL,
    img_id INT NOT NULL,
        FOREIGN KEY (img_id) REFERENCES images(id) ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (ing_group) REFERENCES ingredient_group(ing_g_name) ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT sum_of_macros CHECK (fat_per_100g+protein_per_100g+carbs_per_100g <= 100)
);
CREATE TABLE thematic_section(
        ts_name VARCHAR(100) PRIMARY KEY,
    description VARCHAR(1000) NOT NULL,
    img_id INT NOT NULL,
        FOREIGN KEY (img_id) REFERENCES images(id) ON DELETE RESTRICT ON UPDATE CASCADE
);
CREATE TABLE national_cuisine(
        nt_name VARCHAR(30) PRIMARY KEY,
    img_id INT NOT NULL,
        FOREIGN KEY (img_id) REFERENCES images(id) ON DELETE RESTRICT ON UPDATE CASCADE
);
CREATE TABLE users(
        username VARCHAR(20) PRIMARY KEY,
    password VARCHAR(20) NOT NULL,
    user_type VARCHAR(10) NOT NULL,
    CONSTRAINT password_length CHECK (LENGTH(password) BETWEEN 7 AND 21),
    CONSTRAINT user_type_accepted_values CHECK (user_type in ('admin','chef'))
);
CREATE TABLE chef(
        chef_id INT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    surname VARCHAR(50) NOT NULL,
    username VARCHAR(20) NOT NULL,
    date_of_birth DATE NOT NULL,
    age INT NOT NULL CHECK (age > 0),
        years_of_expertice INT NOT NULL CHECK (years_of_expertice >= 0),
    professional_title VARCHAR(20) NOT NULL,
    img_id INT NOT NULL,
        FOREIGN KEY (img_id) REFERENCES images(id) ON DELETE RESTRICT ON UPDATE CASCADE,
```

```
    FOREIGN KEY (username) REFERENCES users(username) ON DELETE RESTRICT ON UPDATE CASCADE
);
CREATE TABLE admin(
        admin_id INT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    surname VARCHAR(50) NOT NULL,
        username VARCHAR(20) NOT NULL,
    img_id INT NOT NULL,
        FOREIGN KEY (img_id) REFERENCES images(id) ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (username) REFERENCES users(username) ON DELETE RESTRICT ON UPDATE CASCADE
);
CREATE TABLE phonebook(
        phone_number VARCHAR(20) PRIMARY KEY,
        chef_id INT NOT NULL,
        label VARCHAR(30),
        FOREIGN KEY (chef_id) REFERENCES chef(chef_id) ON DELETE RESTRICT ON UPDATE CASCADE
);
CREATE TABLE unit_of_measure(
        u_o_m_id INT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    conversion_rate NUMERIC(8,2) NOT NULL CHECK (conversion_rate >= 0)
);
CREATE TABLE recipe(
        recipe_id INT AUTO_INCREMENT PRIMARY KEY,
        recipe_name VARCHAR(100) NOT NULL,
    description VARCHAR(500) NOT NULL,
    recipe_type VARCHAR(20) NOT NULL,
    difficulty INT NOT NULL CHECK(difficulty >= 1 AND difficulty <= 5),
    prep_time INT NOT NULL CHECK(prep_time >= 0),
    exec_time INT NOT NULL CHECK(exec_time >= 0),
        basic_ingredient INT NOT NULL,
    characterization VARCHAR(100) ,
    national_cuisine VARCHAR(30) NOT NULL,
    portions INT NOT NULL CHECK(portions > 0 ) DEFAULT 0,
    calories_per_portion NUMERIC(8,2) NOT NULL CHECK (calories_per_portion >= 0) DEFAULT 0,
        fat_per_portion NUMERIC(8,2) NOT NULL CHECK (fat_per_portion >= 0) DEFAULT 0,
        protein_per_portion NUMERIC(8,2) NOT NULL CHECK (protein_per_portion >= 0) DEFAULT 0,
        carbs_per_portion NUMERIC(8,2) NOT NULL CHECK (carbs_per_portion >= 0) DEFAULT 0,
    img_id INT NOT NULL,
        FOREIGN KEY (img_id) REFERENCES images(id) ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (national_cuisine) REFERENCES national_cuisine(nt_name) ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (basic_ingredient) REFERENCES ingredient(ing_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT recipe_type_values CHECK (recipe_type in ('savory','confectionery'))
);
CREATE TABLE step(
        step_id INT AUTO_INCREMENT PRIMARY KEY,
    description VARCHAR(1000) NOT NULL,
    step_order INT NOT NULL,
    recipe_id INT NOT NULL,
    FOREIGN KEY (recipe_id) REFERENCES recipe(recipe_id) ON DELETE RESTRICT ON UPDATE CASCADE
);
CREATE TABLE recipe_meal_type(
        rc_id INT NOT NULL,
    mt_name VARCHAR(50) NOT NULL,
    PRIMARY KEY (rc_id,mt_name),
    FOREIGN KEY (rc_id) REFERENCES recipe(recipe_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (mt_name) REFERENCES meal_type(mt_name) ON DELETE RESTRICT ON UPDATE CASCADE
);
CREATE TABLE recipe_label(
        rc_id INT  NOT NULL,
    lb_name VARCHAR(30) NOT NULL,
    PRIMARY KEY (rc_id,lb_name),
    FOREIGN KEY (rc_id) REFERENCES recipe(recipe_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (lb_name) REFERENCES label(label_name) ON DELETE RESTRICT ON UPDATE CASCADE
);
CREATE TABLE recipe_tips(
        rc_id INT NOT NULL,
    tip_id INT NOT NULL,
    PRIMARY KEY (rc_id,tip_id),
```

```
        FOREIGN KEY (rc_id) REFERENCES recipe(recipe_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (tip_id) REFERENCES tip(tip_id) ON DELETE RESTRICT ON UPDATE CASCADE
);
CREATE TABLE equipment_used(
        rc_id INT NOT NULL,
    eq_id INT NOT NULL,
    PRIMARY KEY (rc_id,eq_id),
    FOREIGN KEY (rc_id) REFERENCES recipe(recipe_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (eq_id) REFERENCES equipment(eq_id) ON DELETE RESTRICT ON UPDATE CASCADE
);
CREATE TABLE ingredients_used(
        rc_id INT NOT NULL,
    ing_id INT NOT NULL,
    amount NUMERIC(10,4) NOT NULL CHECK(amount > 0),
    unit INT NOT NULL,
        PRIMARY KEY (rc_id,ing_id),
    FOREIGN KEY (rc_id) REFERENCES recipe(recipe_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (unit) REFERENCES unit_of_measure(u_o_m_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (ing_id) REFERENCES ingredient(ing_id) ON DELETE RESTRICT ON UPDATE CASCADE
);
CREATE TABLE recipe_thematic_section(
        rc_id INT NOT NULL,
    ts_name VARCHAR(100) NOT NULL,
    PRIMARY KEY (rc_id,ts_name),
    FOREIGN KEY (rc_id) REFERENCES recipe(recipe_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (ts_name) REFERENCES thematic_section(ts_name) ON DELETE RESTRICT ON UPDATE CASCADE
);
CREATE TABLE chef_national_cuisines(
        chef_id INT NOT NULL,
    nt_name VARCHAR(30) NOT NULL,
    PRIMARY KEY (chef_id,nt_name),
    FOREIGN KEY (chef_id) REFERENCES chef(chef_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (nt_name) REFERENCES national_cuisine(nt_name) ON DELETE RESTRICT ON UPDATE CASCADE
);
CREATE TABLE chef_recipe(
        chef_id INT NOT NULL,
    rc_id INT NOT NULL,
        PRIMARY KEY (chef_id,rc_id),
    FOREIGN KEY (chef_id) REFERENCES chef(chef_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (rc_id) REFERENCES recipe(recipe_id) ON DELETE RESTRICT ON UPDATE CASCADE
);
CREATE TABLE episode(
        episode_id INT NOT NULL,
    season_id INT NOT NULL,
    img_id INT NOT NULL,
    PRIMARY KEY (episode_id,season_id),
    FOREIGN KEY (img_id) REFERENCES images(id) ON DELETE RESTRICT ON UPDATE CASCADE
);
CREATE TABLE episode_entries(
    entry_id INT AUTO_INCREMENT PRIMARY KEY,
    episode_id INT NOT NULL,
    season_id INT NOT NULL,
    nt_name VARCHAR(30) NOT NULL,
    chef_id INT NOT NULL,
    rc_id INT NOT NULL,
    score1 INT NOT NULL CHECK(score1 >= 1 AND score1 <=5),
        score2 INT NOT NULL CHECK(score2 >= 1 AND score2 <=5),
        score3 INT NOT NULL CHECK(score3 >= 1 AND score3 <=5),
    total_score INT GENERATED ALWAYS AS (score1 + score2 + score3) STORED,
    FOREIGN KEY (episode_id,season_id) REFERENCES episode(episode_id,season_id) ON DELETE RESTRICT ON UPDATE CASCADE,
        FOREIGN KEY (rc_id) REFERENCES recipe(recipe_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (chef_id) REFERENCES chef(chef_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (nt_name) REFERENCES national_cuisine(nt_name) ON DELETE RESTRICT ON UPDATE CASCADE
);
CREATE TABLE judges(
        episode_id INT NOT NULL,
    season_id INT NOT NULL,
    first_judge_id INT NOT NULL,
    second_judge_id INT NOT NULL,
```

```
    third_judge_id INT NOT NULL,
    PRIMARY KEY (episode_id,season_id),
    FOREIGN KEY (episode_id,season_id) REFERENCES episode(episode_id,season_id) ON DELETE RESTRICT ON UPDATE CASCADE,
        FOREIGN KEY (first_judge_id) REFERENCES chef(chef_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (second_judge_id) REFERENCES chef(chef_id) ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (third_judge_id) REFERENCES chef(chef_id) ON DELETE RESTRICT ON UPDATE CASCADE
);

/*------------------------------------------------------------INDEXES-------------------------------------------------------------*/
CREATE INDEX idx_age ON chef(age);

DELIMITER //
/*------------------------------------------------------------PROCEDURES-------------------------------------------------------------*/
CREATE PROCEDURE update_recipe_nutritional_info_after_insert( IN rc_id INT, IN ing_id INT, IN amount NUMERIC(10,4), IN unit INT )
BEGIN
    DECLARE calories NUMERIC(8,2);
    DECLARE fat NUMERIC(8,2);
    DECLARE carbs NUMERIC(8,2);
    DECLARE protein NUMERIC(8,2);

    -- Calculate the nutritional values of the ingredient based on its amount and unit
    SELECT
        (amount * ing.calories_per_100g * uom.conversion_rate / 100),
        (amount * ing.fat_per_100g * uom.conversion_rate / 100),
        (amount * ing.carbs_per_100g * uom.conversion_rate / 100),
        (amount * ing.protein_per_100g * uom.conversion_rate / 100)
    INTO
        calories, fat, carbs, protein
    FROM
        ingredient ing
    JOIN
        unit_of_measure uom ON uom.u_o_m_id = unit
    WHERE
        ing.ing_id = ing_id;

    -- Update the recipe table with the new nutritional values
    UPDATE recipe
    SET
        calories_per_portion = calories_per_portion + calories/portions,
        fat_per_portion = fat_per_portion + fat/portions,
        carbs_per_portion = carbs_per_portion + carbs/portions,
        protein_per_portion = protein_per_portion + protein/portions
    WHERE
        recipe_id = rc_id;
END;
//
CREATE PROCEDURE update_recipe_nutritional_info_after_delete( IN rc_id INT, IN ing_id INT, IN amount NUMERIC(10,4), IN unit INT)
BEGIN
    DECLARE calories NUMERIC(8,2);
    DECLARE fat NUMERIC(8,2);
    DECLARE carbs NUMERIC(8,2);
    DECLARE protein NUMERIC(8,2);

    -- Calculate the nutritional values of the ingredient based on its amount and unit
    SELECT
        (amount * ing.calories_per_100g * uom.conversion_rate / 100),
        (amount * ing.fat_per_100g * uom.conversion_rate / 100),
        (amount * ing.carbs_per_100g * uom.conversion_rate / 100),
        (amount * ing.protein_per_100g * uom.conversion_rate / 100)
    INTO
        calories, fat, carbs, protein
    FROM
        ingredient ing
    JOIN
        unit_of_measure uom ON uom.u_o_m_id = unit
    WHERE
        ing.ing_id = ing_id;

    -- Update the recipe table by subtracting the nutritional values of the deleted ingredient
```

```
287     UPDATE recipe
288     SET
289         calories_per_portion = calories_per_portion - calories/portions,
290         fat_per_portion = fat_per_portion - fat/portions,
291         carbs_per_portion = carbs_per_portion - carbs/portions,
292         protein_per_portion = protein_per_portion - protein/portions
293     WHERE
294         recipe_id = rc_id;
295 END;
296 //
297 CREATE PROCEDURE GetChefsByCuisineAndSeason(IN cuisine VARCHAR(255), IN season INT)
298 BEGIN
299         SELECT DISTINCT chef.chef_id,chef.name,chef.surname
300         FROM episode_entries
301         INNER JOIN chef
302         ON chef.chef_id = episode_entries.chef_id
303         WHERE episode_entries.nt_name = cuisine;
304 
305         SELECT DISTINCT chef.chef_id,chef.name,chef.surname
306         FROM episode_entries
307         INNER JOIN chef
308         ON chef.chef_id = episode_entries.chef_id
309         WHERE episode_entries.nt_name = cuisine
310         AND episode_entries.season_id = season
311         ORDER BY chef_id;
312 END;
313 //
314 /*------------------------------------------------------------TRIGERS-------------------------------------------------------------------*/
315 CREATE TRIGGER update_nutritional_info_after_insert_t AFTER INSERT ON ingredients_used
316 FOR EACH ROW
317 BEGIN
318     CALL update_recipe_nutritional_info_after_insert(NEW.rc_id, NEW.ing_id, NEW.amount, NEW.unit);
319 END;
320 //
321 CREATE TRIGGER update_nutritional_info_after_delete_t AFTER DELETE ON ingredients_used
322 FOR EACH ROW
323 BEGIN
324         CALL update_recipe_nutritional_info_after_delete(OLD.rc_id, OLD.ing_id, OLD.amount, OLD.unit);
325 END;
326 //
327 CREATE TRIGGER update_nutritional_info_after_update_t AFTER UPDATE ON ingredients_used
328 FOR EACH ROW
329 BEGIN
330         CALL update_recipe_nutritional_info_after_delete(OLD.rc_id, OLD.ing_id, OLD.amount, OLD.unit);
331     CALL update_recipe_nutritional_info_after_insert(NEW.rc_id, NEW.ing_id, NEW.amount, NEW.unit);
332 END;
333 //
334 CREATE TRIGGER autocharacterization BEFORE INSERT ON recipe
335 FOR EACH ROW
336 BEGIN
337     DECLARE rec_character VARCHAR(100);
338     SELECT characterization INTO rec_character
339     FROM ingredient
340     INNER JOIN ingredient_group ON ingredient.ing_group = ingredient_group.ing_g_name
341     WHERE ing_id = NEW.basic_ingredient;
342 
343     SET NEW.characterization = rec_character;
344 END//
345 CREATE TRIGGER invalid_years_of_expertice BEFORE INSERT ON chef
346 FOR EACH ROW
347 BEGIN
348     IF (NEW.age < NEW.years_of_expertice) THEN
349     SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Years of expertice cannot be greater than the age of the chef';
350         END IF;
351 END//
352 CREATE TRIGGER consecutive_episodes_check BEFORE INSERT ON episode_entries
353 FOR EACH ROW
354 BEGIN
355 
```

```
356     -- Check if the chef has participated in the last three episodes
357     IF EXISTS (
358           SELECT 1
359           FROM episode_entries
360           WHERE chef_id = NEW.chef_id AND season_id = NEW.season_id AND episode_id IN (new.episode_id, new.episode_id - 1, new.
                    episode_id - 2)
361           GROUP BY chef_id, season_id
362           HAVING COUNT(*) >= 3
363        ) THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Chef cannot compete in more than 3 consecutive episodes in the same
                season.';
364     END IF;
365
366     -- Check if the recipe has participated in the last three episodes
367     IF EXISTS (
368           SELECT 1
369           FROM episode_entries
370           WHERE rc_id = NEW.rc_id AND season_id = NEW.season_id AND episode_id IN (new.episode_id, new.episode_id - 1, new.
                    episode_id - 2)
371           GROUP BY rc_id, season_id
372           HAVING COUNT(*) >= 3
373        ) THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Recipe cannot compete in more than 3 consecutive episodes in the same
                season.';
374     END IF;
375
376     -- Check if the national cuisine has participated in the last three episodes
377     IF EXISTS (
378           SELECT 1
379           FROM episode_entries
380           WHERE nt_name = NEW.nt_name AND season_id = NEW.season_id AND episode_id IN (new.episode_id, new.episode_id - 1, new.
                    episode_id - 2)
381           GROUP BY nt_name, season_id
382           HAVING COUNT(*) >= 3
383        ) THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'A national cuisine cannot compete in more than 3 consecutive episodes
                in the same season.';
384     END IF;
385 END//
386 CREATE TRIGGER consecutive_judges_check BEFORE INSERT ON judges
387 FOR EACH ROW
388 BEGIN
389
390     -- Check if the first judge has participated in the last three episodes
391     IF EXISTS (
392           SELECT 1
393           FROM judges
394           WHERE (first_judge_id = NEW.first_judge_id OR second_judge_id = NEW.first_judge_id OR third_judge_id = NEW.
                    first_judge_id ) AND
395           season_id = NEW.season_id AND episode_id IN (new.episode_id, new.episode_id - 1, new.episode_id - 2)
396           GROUP BY season_id
397           HAVING COUNT(*) >= 3) THEN
398           SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Judge cannot compete in more than 3 consecutive episodes in the same
                    season.';
399     END IF;
400
401     -- Check if the second judge has participated in the last three episodes
402     IF EXISTS (
403           SELECT 1
404           FROM judges
405           WHERE (first_judge_id = NEW.second_judge_id OR second_judge_id = NEW.second_judge_id OR third_judge_id = NEW.
                    second_judge_id ) AND
406           season_id = NEW.season_id AND episode_id IN (new.episode_id, new.episode_id - 1, new.episode_id - 2)
407           GROUP BY season_id
408           HAVING COUNT(*) >= 3
409        ) THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Judge cannot compete in more than 3 consecutive episodes in the same
                season.';
410     END IF;
411
412     -- Check if the third judge has participated in the last three episodes
413     IF EXISTS (
414           SELECT 1
```

```
415             FROM judges
416             WHERE (first_judge_id = NEW.third_judge_id OR second_judge_id = NEW.third_judge_id OR third_judge_id = NEW.
                    third_judge_id ) AND
417             season_id = NEW.season_id AND episode_id IN (new.episode_id, new.episode_id - 1, new.episode_id - 2)
418             GROUP BY season_id
419             HAVING COUNT(*) >= 3
420         ) THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Judge cannot compete in more than 3 consecutive episodes in the same
                season.';
421     END IF;
422 END//
423 CREATE TRIGGER check_tip_recipe_limit BEFORE INSERT ON recipe_tips
424 FOR EACH ROW
425 BEGIN
426     DECLARE tip_count INT;
427
428     -- Count the current number of tips for the recipe
429     SELECT COUNT(*)
430     INTO tip_count
431     FROM recipe_tips
432     WHERE rc_id = NEW.rc_id;
433
434     -- Check if the number of tips exceeds the limit
435     IF tip_count >= 3 THEN
436         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'A recipe cannot have more than 3 tips.';
437     END IF;
438 END//
439
440 DELIMITER ;
```

Listing 1: DDL Script

8. DML script

Το DML(Data Manipulation Language) script αφορά την εισαγωγή δεδμομένων στη βάση, τη δημιουργία όψεων (views) κ.α.. Το DML script αποτελείται από κώδικα 3600 γραμμών, οπότε δεν θα ήταν εφικτό να παρατεθεί αυτούσιο στην αναφορά για αυτό το λόγο παρατίθεται παρακάτω ένα κομμάτι από το DML script για την δική μας βάση cooking_competition που αφορά την εισαγωγή των δεδομένων για τον πίνακα ingredien_group:

```
1  INSERT INTO ingredient_group (ing_g_name, description, characterization, img_id) VALUES
2  ('Grains', 'Staple foods like wheat, rice, and oats, providing carbohydrates and fiber.', 'Grain-based', '50'),
3  ('Milk and milk products', 'Dairy items such as milk, cheese, and yogurt, rich in calcium and protein.', 'Dairy', '51'),
4  ('Fruit and fruit products', 'Fresh fruits and their derivatives, offering vitamins, fiber, and antioxidants.', 'Fruity', '52'),
5  ('Eggs', 'Versatile protein source, packed with essential nutrients.', 'Egg-centric', '53'),
6  ('Meat and poultry', 'Animal-derived foods like beef and chicken, high in protein and iron.', 'Meat-focused', '54'),
7  ('Fish/shellfish', 'Seafood options rich in omega-3 fatty acids and lean protein.', 'Seafood', '55'),
8  ('Vegetables', 'Nutrient-dense plant foods like spinach and carrots, offering vitamins and minerals.', 'Plant-based', '56'),
9  ('Fats/oils', 'Cooking essentials like olive oil and butter, providing energy and flavor.', 'Oily', '57'),
10 ('Legumes/nuts/seeds', 'Plant-based protein sources such as beans, almonds, and chia seeds.', 'Nutty', '58'),
11 ('Sugar and sugar products', 'Sweeteners like cane sugar and honey, adding sweetness to foods.', 'Dessert-inspired', '59'),
12 ('Non-alcoholic beverages', 'Refreshing drinks such as water, tea, and coffee, hydrating and flavorful.', 'Beverage-based','60'),
13 ('Alcoholic beverages', 'Drinks containing alcohol, including beer, wine, and spirits, enjoyed responsibly for social and
       culinary purposes.', 'Cocktail-infused', '61'),
14 ('Herbs/Spices', 'A diverse array of plant-derived ingredients, offering rich flavors and aromas to elevate culinary creations.',
        'Aromatic', '62');
```

Listing 2: DML Script (for table ingredient_group)

Views

Τα Views είναι εικονικοί πίνακες που δημιουργούνται με τη βοήθεια κάποιων query και αποθηκεύονται στη βάση για μετέπειτα χρήση, χωρίς όμως να αποθηκεύουν δεδομένα.
Στη βάση μας εισάγαμε ένα view το οποίο περιέχει όλα τα άτομα που συμμετείχαν είτε ως μάγειρες είτε ως κριτές σε κάθε επεισόδειο. Η δημιουργία του query αυτού βρίσκεται στο DML script,.

```sql
CREATE VIEW chefs_in_episodes (episode_id, season_id, chef_id, job) AS
        SELECT episode_id, season_id, chef_id, "chef" as job
        FROM episode_entries
        UNION ALL
        SELECT episode_id, season_id, first_judge_id as chef_id, "judge" as job
        FROM judges
        UNION ALL
        SELECT episode_id, season_id, second_judge_id as chef_id, "judge" as job
        FROM judges
        UNION ALL
        SELECT episode_id, season_id, third_judge_id as chef_id, "judge" as job
        FROM judges;
```

Listing 3: DML Script (Views)

9. Random Episode Generator

Για τους σκοπούς της εργασίας ζητήθηκε από εμάς τα επεισόδια να κληρώνονται τυχαία με 10 επεισόδια ανά σεζόν και 10 συμμετοχές σε κάθε επεισόδιο, με βάση κάποιους περιορισμούς. Για την υλοποίηση του συγκεκριμένου μέρους της εργασίας, χρησιμοποιήσαμε Node JS, όπου γράψαμε 2 κώδικες με ονόματα random.js kai random_judges.js τα οποία εισάγουν τυχαία δεδομένα στους πίνακες episode_entries και judges αντίστοιχα, με βάση κάποιους περιορισμούς, για 5 σεζόν.
Οι εν λόγω κώδικες παρατίθενται παρακάτω:

```javascript
const mysql = require('mysql');

// Create MySQL connection
const connection = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: 'password',
    database: 'cooking_competition'
});

// Connect to MySQL
connection.connect((err) => {
    if (err) {
        console.error('Error connecting to database: ' + err.stack);
        return;
    }

    console.log('Connected to database');


    // Define async function to generate data
    const generateData = async () => {
        try {
            // Define async function to generate data for a single season
            const generateDataForSeason = async (season,nationalCuisines) => {
                // Fetch participation in previous episodes
```

```
const previousEpisodesChef = {};
const previousEpisodesNt = {};
const previousEpisodesRc = {};
// Function to generate data for a single episode
const generateDataForEpisode = async (episode, previousEpisodesChef, previousEpisodesRc, previousEpisodesNt) => {
    console.log(`Generating data for Season ${season}, Episode ${episode}`);

    const episodeCuisines = [];

    // List to keep track of selected chefs for this episode
    const selectedChefs = [];

    // Loop until we have selected 10 unique cuisines for this episode
    while (episodeCuisines.length < 10) {
        // Find a valid national cuisine not participating in the last 3 episodes
        const validCuisine = await findValidCuisine(previousEpisodesNt, episode, nationalCuisines);

        // Check if the cuisine is not already selected for this episode
        if (!episodeCuisines.includes(validCuisine.nt_name)) {
            episodeCuisines.push(validCuisine.nt_name);

            let chef_id;
            let recipe;

            // Attempt to find a valid chef and recipe until successful or exhausted
            let attempts = 0;
            while (attempts < 25) {
                // Find a valid chef for the cuisine not participating in the last 3 episodes
                chef_id = await findValidChef(validCuisine.nt_name, previousEpisodesChef, episode, selectedChefs)
                      ;

                // Find a valid recipe for the cuisine and chef not participating in the last 3 episodes
                recipe = await findValidRecipe(validCuisine.nt_name, chef_id, previousEpisodesRc, episode);

                // If a valid recipe is found, break the loop
                if (recipe) {
                    break;
                }

                // Increment attempts and try again with another chef
                attempts++;
            }

            // Output episode data
            console.log(`Season ${season}, Episode ${episode}, Entry ${(episode - 1) * 10 + (season - 1) * 100 +
                  episodeCuisines.length}:`);
            console.log(`National Cuisine: ${validCuisine.nt_name}`);
            console.log(`Recipe: ${recipe ? recipe.recipe_name : 'No valid recipe found'}`);
            console.log(`Chef: ${chef_id}`);
            console.log('----------------------');

            // Insert episode data into episode_entries table
            const episodeEntry = {
                entry_id: (episode - 1) * 10 + (season - 1) * 100 + episodeCuisines.length,
                episode_id: episode,
                season_id: season,
                nt_name: validCuisine.nt_name,
                chef_id: chef_id,
                rc_id: recipe ? recipe.recipe_id : null,
                score1: generateRandomScore(),
                score2: generateRandomScore(),
                score3: generateRandomScore()
            };

            await new Promise((resolve, reject) => {
                connection.query('INSERT INTO episode_entries SET ?', episodeEntry, (err, result) => {
                    if (err) reject(err);
                    console.log(`Inserted episode entry with ID ${result.insertId}`);
                    resolve();
```

```
                            });
                        });

                        // Add selected chef to the list
                        selectedChefs.push(chef_id);
                        if (!previousEpisodesChef[chef_id]) {
                                previousEpisodesChef[chef_id] = [];
                        }
                        if (!previousEpisodesRc[recipe.recipe_id]) {
                            previousEpisodesRc[recipe.recipe_id] = [];
                        }
                        if (!previousEpisodesNt[nationalCuisines.indexOf(validCuisine.nt_name)]) {
                            previousEpisodesNt[nationalCuisines.indexOf(validCuisine.nt_name)] = [];
                        }
                        previousEpisodesChef[chef_id].push(episode);
                        previousEpisodesRc[recipe.recipe_id].push(episode);
                        previousEpisodesNt[nationalCuisines.indexOf(validCuisine.nt_name)].push(episode);


                    }
                }
            };


            // Generate data for each episode of the current season
            for (let episode = 1; episode <= 10; episode++) {
                await generateDataForEpisode(episode, previousEpisodesChef, previousEpisodesRc,previousEpisodesNt);
            }
        };

        // Function to find a valid national cuisine not participating in the last 3 episodes
        const findValidCuisine = async (previousEpisodes, episode, nationalCuisines) => {
            let cuisine;
            let selectedCuisine = false;

            while (!selectedCuisine) {
                cuisine = await new Promise((resolve, reject) => {
                    connection.query('SELECT * FROM national_cuisine ORDER BY RAND() LIMIT 1', (err, result) => {
                    if (err) reject(err);
                        resolve(result[0]);
                    });
                });

                // Check if the cuisine has participated in the last 3 episodes
                const cuisineName = cuisine.nt_name;
                if (!previousEpisodes[nationalCuisines.indexOf(cuisineName)] || !previousEpisodes[nationalCuisines.indexOf(
                      cuisineName)].includes(episode - 1) || !previousEpisodes[nationalCuisines.indexOf(cuisineName)].
                      includes(episode - 2) || !previousEpisodes[nationalCuisines.indexOf(cuisineName)].includes(episode - 3)
                      ) {
                    selectedCuisine = true;
                }
            }

            return cuisine;
        };

        const findValidChef = async (cuisineName, previousEpisodes, episode, selectedChefs) => {
            let chef_id;
            let selectedChef = false;

            while (!selectedChef) {
                // Select a random chef for the cuisine
                const chefResult = await new Promise((resolve, reject) => {
                    connection.query('SELECT * FROM chef_national_cuisines INNER JOIN chef ON chef_national_cuisines.chef_id
                          = chef.chef_id WHERE nt_name = ? ORDER BY RAND() LIMIT 1', [cuisineName], (err, result) => {
                        if (err) reject(err);
                        resolve(result);
                    });
                });
```

```
159
160                     chef_id = chefResult[0].chef_id;
161
162                     // Check if the chef has participated in the previous 3 episodes and is not already selected for this episode
163                     if ((!previousEpisodes[chef_id] || !previousEpisodes[chef_id].includes(episode - 1) || !previousEpisodes[
                            chef_id].includes(episode - 2) || !previousEpisodes[chef_id].includes(episode - 3)) && !selectedChefs.
                            includes(chef_id)) {
164                         selectedChef = true;
165                     }
166                 }
167
168                 return chef_id;
169             };
170
171             // Function to find a valid recipe not participating in the last 3 episodes
172             const findValidRecipe = async (cuisine, chef_id, previousEpisodes, episode) => {
173                 let recipe;
174                 let selectedRecipe = false;
175
176                 while (!selectedRecipe) {
177                     recipe = await new Promise((resolve, reject) => {
178                         connection.query('SELECT * FROM recipe INNER JOIN chef_recipe ON recipe.recipe_id = chef_recipe.rc_id
                                WHERE recipe.national_cuisine = ? AND chef_recipe.chef_id = ? ORDER BY RAND() LIMIT 1', [cuisine,
                                chef_id], (err, result) => {
179                             if (err) reject(err);
180                             resolve(result[0]);
181                         });
182                     });
183
184                     // Check if the recipe has participated in the last 3 episodes
185                     const recipeId = recipe ? recipe.recipe_id : null;
186                     if (!previousEpisodes[recipeId] || !previousEpisodes[recipeId].includes(episode - 1) || !previousEpisodes[
                            recipeId].includes(episode - 2) || !previousEpisodes[recipeId].includes(episode - 3)) {
187                         selectedRecipe = true;
188                     }
189                 }
190
191                 return recipe;
192             };
193
194             // Function to generate random score from 1 to 5
195             const generateRandomScore = () => {
196                 return Math.floor(Math.random() * 5) + 1;
197             };
198
199             // Generate data for 5 seasons
200             for (let season = 1; season <= 5; season++) {
201                 console.log(`Generating data for Season ${season}`);
202
203                 // Fetch all cuisines
204                 const nationalCuisines = await new Promise((resolve, reject) => {
205                     connection.query('SELECT nt_name FROM national_cuisine', (err, results) => {
206                         if (err) reject(err);
207                         const cuisineNames = results.map(row => row.nt_name);
208                         resolve(cuisineNames);
209                     });
210                 });
211
212                 await generateDataForSeason(season, nationalCuisines);
213             }
214         } catch (error) {
215             console.error('Error generating data: ' + error);
216         } finally {
217             // Close MySQL connection after generating data for all seasons
218             connection.end();
219         }
220     };
221
222     // Call the async function to generate data
```

```
    generateData();
});
```

Listing 4: random.js Script

```
const mysql = require('mysql');

// Create MySQL connection
const connection = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: 'password',
    database: 'cooking_competition'
});

// Connect to MySQL
connection.connect((err) => {
    if (err) {
        console.error('Error connecting to database: ' + err.stack);
        return;
    }
    console.log('Connected to database');

    // Function to shuffle an array
    const shuffle = (array) => {
        for (let i = array.length - 1; i > 0; i--) {
            const j = Math.floor(Math.random() * (i + 1));
            [array[i], array[j]] = [array[j], array[i]];
        }
        return array;
    };

    // Define async function to generate judges for each episode
    const generateJudgesForEpisode = async (episode, season, previousJudges) => {
        console.log(`Generating judges for Season ${season}, Episode ${episode}`);

        // Retrieve random chefs who are eligible to be judges
        const eligibleJudges = await new Promise((resolve, reject) => {
            connection.query('SELECT * FROM chef WHERE NOT EXISTS (SELECT * FROM episode_entries WHERE episode_id = ? AND
                  season_id = ? AND chef_id = chef.chef_id) ORDER BY RAND() LIMIT 3', [episode, season], (err, result) => {
                if (err) reject(err);
                resolve(result);
            });
        });

        // Shuffle the eligible judges randomly
        const shuffledJudges = shuffle(eligibleJudges);

        // Select the first three judges
        const judges = shuffledJudges.slice(0, 3);

        // Output judges data
        console.log(`Judges for Season ${season}, Episode ${episode}:`);
        console.log(`First Judge: ${judges[0].chef_id}`);
        console.log(`Second Judge: ${judges[1].chef_id}`);
        console.log(`Third Judge: ${judges[2].chef_id}`);
        console.log('----------------------');

        // Insert judges data into the judges table
        const judgesEntry = {
            episode_id: episode,
            season_id: season,
            first_judge_id: judges[0].chef_id,
            second_judge_id: judges[1].chef_id,
            third_judge_id: judges[2].chef_id
        };

        await new Promise((resolve, reject) => {
```

```
            connection.query('INSERT INTO judges SET ?', judgesEntry, (err, result) => {
                if (err) reject(err);
                console.log(`Inserted judges entry for Episode ${episode} of Season ${season}`);
                resolve();
            });
        });
    };

    // Define async function to generate judges for each season
    const generateJudgesForSeason = async (season) => {
        // Fetch participation of judges in previous episodes
        const previousJudges = {};
        const judges = await new Promise((resolve, reject) => {
            connection.query('SELECT * FROM judges WHERE season_id = ?', [season], (err, result) => {
                if (err) reject(err);
                resolve(result);
            });
        });

        judges.forEach((judge) => {
            const episode_id = judge.episode_id;
            if (!previousJudges[episode_id]) {
                previousJudges[episode_id] = [];
            }
            previousJudges[episode_id].push(judge.first_judge_id);
            previousJudges[episode_id].push(judge.second_judge_id);
            previousJudges[episode_id].push(judge.third_judge_id);
        });

        // Generate judges for each episode of the current season
        for (let episode = 1; episode <= 10; episode++) {
            await generateJudgesForEpisode(episode, season, previousJudges);
        }
    };

    // Define async function to generate judges for all seasons
    const generateJudges = async () => {
        for (let season = 1; season <= 5; season++) {
            console.log(`Generating judges for Season ${season}`);
            await generateJudgesForSeason(season);
        }
    };

    // Call the async function to generate judges
    generateJudges();
});
```

Listing 5: random_judges.js Script

10. Permissions

Σε αυτό το μέρος της εργασίας, δημιουργούμε χρήστες για κάθε ένα από τους μάγειρες(chef) και τους διαχειριστές (admin) της βάσης και τους επιτρέπουμε να εκτελούν κάποιες ενέργειες στη βάση. Πιο συγκεκριμένα:

- Ο Διαχειριστής μπορεί να καταχωρεί και να τροποποιεί όλα τα δεδομένα, να δημιουργεί αντίγραφο ασφαλείας της βάσης και να επαναφέρει το σύστημα από αυτό.
- Ο Μάγειρας μπορεί να τροποποιεί τα προσωπικά του στοιχεία.
- Ο Μάγειρας μπορεί να εισάγει καινούρια συνταγή.

- Ο Μάγειρας μπορεί να τροποποιεί συνταγές τις οποίες έχει ανατεθεί.

Ο κώδικας για αυτό το μέρος της εργασίας βρίσκεται στο αρχείο roles.sql το οποίο περιέχει τα εξής:

- Stored procedure update_chef_info
  Μια διαδικασία για την ενημέρωση των προσωπικών στοιχείων ενός συγκεκριμένου μάγειρα που είναι ο τρέχων χρήστης.
- Stored procedure update_recipe
  Μια διαδικασία για την ενημέρωση των στοιχείων μιας συνταγής ενός συγκεκριμένου μάγειρα που είναι ο τρέχων χρήστης.
- Δημιουργία του ρόλου chef_role και αδειοδότηση για εκτέλεση των παραπάνω διαδικασιών και εισαγωγή καινούριας συνταγής.
- Stored procedure create_chef_users
  Μια διαδικασία για την δημιουργία ενός χρήστη για κάθε μάγειρα με username το username του μάγειρα και identifier το password του και ανάθεση σε αυτούς του ρόλου chef_role
- Δημιουργία χρηστών για τους 2 διαχειριστές και ανάθεση όλων των δικαιωμάτων σε αυτούς.

Παρατίθεται παρακάτω ο κώδικας από το συγκεκριμένο αρχείο:

```
DELIMITER //
-- Procedure to update chef information
CREATE PROCEDURE update_chef_info(IN id INT, IN new_name VARCHAR(50),  IN new_surname VARCHAR(50), IN new_date_of_birth date, IN
      new_age INT,IN new_years_of_expertice INT,IN new_professional_title VARCHAR(20), IN new_img_id INT)
BEGIN
    IF (id = CURRENT_USER_ID()) THEN
        UPDATE chef SET name = new_name, surname = new_surname, _date_of_birth = new_date_of_birth, age = new_age,
        years_of_expertice = new_years_of_expertice, professional_title = new_professional_title, img_id = new_img_id WHERE id =
              chef_id;
    ELSE
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Permission denied';
    END IF;
END //


-- Procedure to update a recipe
CREATE PROCEDURE update_recipe(IN recip_id INT, IN c_id INT, IN new_name VARCHAR(100), IN new_description VARCHAR(500),  IN
      new_recipe_type VARCHAR(20), IN new_difficulty INT, IN new_ INT,
IN new_exec_time INT, IN new_basic_ingredient INT,IN new_nt VARCHAR(30), IN new_portions INT, IN new_img_id INT)
BEGIN
    IF ( (EXISTS(SELECT * FROM chef_recipe WHERE (chef_id = c_id AND rc_id = recip_id)))  AND c_id = CURRENT_USER_ID()) THEN
        UPDATE recipe SET name = new_name, description = new_description, recipe_type = new_recipe_type, difficulty =
              new_difficulty, prep_time = new_prep_time,
        exec_time = new_exec_time, basic_ingredient = new_basic_ingredient, national_cuisine = new_nt, portions = new_portions,
              img_id = new_img_id WHERE recipe_id = recip_id;
    ELSE
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Permission denied';
    END IF;
END //
DELIMITER ;
CREATE ROLE chef_role;
GRANT EXECUTE ON PROCEDURE update_chef_info TO chef_role;
GRANT EXECUTE ON PROCEDURE update_recipe TO chef_role;
GRANT INSERT ON cooking_competition.recipe  TO chef_role;
GRANT INSERT ON cooking_competition.ingredients_used TO chef_role;
```

```sql
GRANT INSERT ON cooking_competition.equipment_used TO chef_role;
GRANT INSERT ON cooking_competition.recipe_label TO chef_role;
GRANT INSERT ON cooking_competition.recipe_meal_type TO chef_role;
GRANT INSERT ON cooking_competition.recipe_thematic_section TO chef_role;
GRANT INSERT ON cooking_competition.recipe_tips TO chef_role;
GRANT INSERT ON cooking_competition.step TO chef_role;
DELIMITER //
CREATE PROCEDURE create_chef_users()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE chef_id INT;
    DECLARE chef_username VARCHAR(255);
    DECLARE chef_password VARCHAR(255);
    DECLARE cur CURSOR FOR SELECT c.chef_id, c.username, u.password FROM chef c INNER JOIN users u ON c.username = u.username;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur;

    read_loop: LOOP
        FETCH cur INTO chef_id, chef_username, chef_password;
        IF done THEN
            LEAVE read_loop;
        END IF;

        -- Dynamic SQL to create user
        SET @sql = CONCAT('CREATE USER "', chef_username, '"@"localhost" IDENTIFIED BY "', chef_password, '";');
        PREPARE stmt FROM @sql;
        EXECUTE stmt;
        DEALLOCATE PREPARE stmt;

        -- Grant necessary permissions to the user
        SET @grant_sql = CONCAT('GRANT chef_role TO "', chef_username, '"@"localhost";');
        PREPARE grant_stmt FROM @grant_sql;
        EXECUTE grant_stmt;
        DEALLOCATE PREPARE grant_stmt;
    END LOOP;

    CLOSE cur;
END //
DELIMITER ;
CREATE USER 'admin_1'@'localhost' IDENTIFIED BY 'W3lc0m3';
CREATE USER 'admin_2'@'localhost' IDENTIFIED BY 'I4mth3b3st';
GRANT ALL PRIVILEGES ON cooking_competition.* TO 'admin_1'@'localhost';
GRANT ALL PRIVILEGES ON cooking_competition.* TO 'admin_2'@'localhost';
CALL create_chef_users;
```

Listing 6: roles.sql

11. Queries-Ερωτήματα

Στην παράγραφο αυτή παρουσιάζονται τα ερωτήματα που κληθήκαμε να απαντήσουμε με τη βοήθεια της SQL και για τα οποία παρουσιάζουμε την εκφώνηση, τυχόν παραδοχές που χρειάζονται, τον κώδικα σε SQL του ερωτήματος και το αποτέλεσμα που μας επιστρέφει.

11.1

Εκφώνηση:

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

Κώδικας:

```sql
SELECT chef.chef_id,name,surname, avg(total_score/3) average_score
FROM episode_entries
INNER JOIN chef
ON episode_entries.chef_id = chef.chef_id GROUP BY chef.chef_id;

SELECT nt_name, avg(total_score/3) average_score
FROM episode_entries
GROUP BY nt_name;
```

Listing 7: Query 1

Αποτελέσματα:

chef_id	name	surname	average_score
1	Lefteris	Lazarou	2.77777778
2	Yiannis	Lucacos	2.91666667
3	Dimitris	Skarmoutsos	3.16666667
4	Sotiris	Kontizas	3.03333333
5	Panos	Ioannidis	3.09090909
6	Leonidas	Koutsopoulos	2.96969697
7	Margarita	Nikolaidi	2.95833333
8	Giovanni	Scaraggi	4.00000000
9	Akis	Petretzikis	2.83333333
10	Christos	Glossidis	3.16666667
11	Giorgos	Porfiris	2.77777778
12	Stavros	Varthalitis	2.83333333
13	Stavros	Georgiou	3.07407407
14	Spiridoula	Karampoutaki	3.00000000
15	Ilias	Kaziolis	2.94444444
16	Maria	Bay	3.22222222
17	Ioannis	Charalampous	3.06666667
18	Dimitris	Bellos	2.38095238
19	John	Reano	2.93939394
20	Maria	Lazaridou	2.85185185
21	Panagiotis	Tzamalis	2.73333333
22	Pavlos	Happilos	2.77777778
23	Stefanos	Hilas	3.44444444
24	Grhgorhs	Giannopoulos	3.40740741
25	Nikos	Trakas	2.77777778
26	Panagiotis	Avgenikos	3.28571429
27	Vaggelis	Filaktakis	2.66666667
28	Dimitris	Politakis	2.60000000
29	Giannis	Chatzikas	2.83333333
30	Nikitas	Filippakis	3.19047619
30	Nikitas	Filippakis	3.19047619
31	Konstantinos	Kokkotas	2.83333333
32	Alexandros	Antonelakis	3.12820513
33	Alkistis	Alexaki	2.86111111
34	Asimina	Oustalli	2.97222222
35	Giorgos	Orfanos	3.06666667
36	Zahir	Yawary	3.76190476
37	Manos	Sarris	2.42424242
38	Panos	Togias	2.50000000
39	Padelis	Vouros	3.00000000
40	Stamatis	Kovaios	3.20000000
41	Tasos	Kiriklakis	3.30303030
42	Charalambos	Kotsonis	3.00000000
43	Xristos	Barkas	3.22222222
44	Giorgos	Chronakis	3.33333333
45	Gogo	Delogianni	3.12500000
46	Dimitris	Papapanagiotou	3.16666667
47	Zachos	Papadakis	2.75757576
48	Lambros	Vakiaros	3.15151515
49	Levis	Kodhima	3.14285714
50	Ektoras	Botrini	3.10000000
51	Adam	Kontovas	3.16666667
52	Athinagoras	Kostakos	2.90476190
53	Aris	Vezenes	3.60000000
54	Nikos	Thomas	3.24242424
55	Christina	Christophi	3.08333333
56	Ilias	Panagiotou	2.88888889
57	Lefteris	Zafeiropoulos	3.18518518
58	Marios	Ioannou	2.85185185
59	Christos	Moiras	3.18518518
60	Kostas	Fretzios	3.06666667

nt_name	average_score
American	2.79166667
Arab	3.25396825
Australian	2.97101449
Brazilian	3.01234568
British	2.77777778
Chinese	3.14583333
Cypriot	2.90476190
Egyptian	3.06349206
French	2.83870968
German	2.92424242
Greek	2.94666667
Indian	2.95833333
Italian	3.06944444
Japanese	3.04938272
Korean	3.32098765
Lebanese	3.04545455
Mexican	3.15942029
Polish	3.01234568
Russian	3.10256410
South African	2.98245614
Spanish	3.02564103

11.2

Εκφώνηση:

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

Παραδοχές:

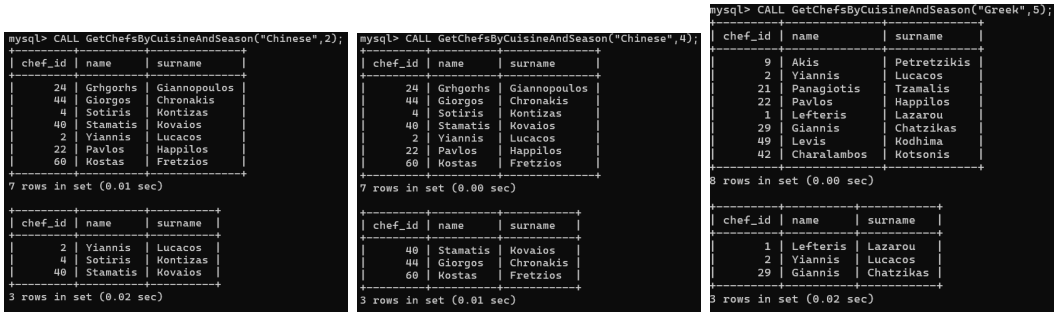
Για την εκτέλεση του συγκεκριμένου ερωτήματος δημιουργήσαμε μια διαδικασία την οποία περιγράψαμε παραπάνω έτσι ώστε να μπορούμε να εκτελούμε το ερώτημα για διάφορους συνδυασμούς εθνικής κουζίνας και έτους, ενδεικτικά παρουσιάζουμε 3 παραδείγματα.

Κώδικας:

```
CALL GetChefsByCuisineAndSeason("Chinese",2);
CALL GetChefsByCuisineAndSeason("Chinese",4);
CALL GetChefsByCuisineAndSeason("Greek",5);
```

Listing 8: Query 2

Αποτελέσματα:



```
mysql> CALL GetChefsByCuisineAndSeason("Chinese",2);
+---------+----------+--------------+
| chef_id | name     | surname      |
+---------+----------+--------------+
|      24 | Grhgorhs | Giannopoulos |
|      44 | Giorgos  | Chronakis    |
|       4 | Sotiris  | Kontizas     |
|      40 | Stamatis | Kovaios      |
|       2 | Yiannis  | Lucacos      |
|      22 | Pavlos   | Happilos     |
|      60 | Kostas   | Fretzios     |
+---------+----------+--------------+
7 rows in set (0.01 sec)

+---------+----------+----------+
| chef_id | name     | surname  |
+---------+----------+----------+
|       2 | Yiannis  | Lucacos  |
|       4 | Sotiris  | Kontizas |
|      40 | Stamatis | Kovaios  |
+---------+----------+----------+
3 rows in set (0.02 sec)

mysql> CALL GetChefsByCuisineAndSeason("Chinese",4);
+---------+----------+--------------+
| chef_id | name     | surname      |
+---------+----------+--------------+
|      24 | Grhgorhs | Giannopoulos |
|      44 | Giorgos  | Chronakis    |
|       4 | Sotiris  | Kontizas     |
|      40 | Stamatis | Kovaios      |
|       2 | Yiannis  | Lucacos      |
|      22 | Pavlos   | Happilos     |
|      60 | Kostas   | Fretzios     |
+---------+----------+--------------+
7 rows in set (0.00 sec)

+---------+----------+----------+
| chef_id | name     | surname  |
+---------+----------+----------+
|      40 | Stamatis | Kovaios  |
|      44 | Giorgos  | Chronakis |
|      60 | Kostas   | Fretzios |
+---------+----------+----------+
3 rows in set (0.01 sec)

mysql> CALL GetChefsByCuisineAndSeason("Greek",5);
+---------+-------------+-------------+
| chef_id | name        | surname     |
+---------+-------------+-------------+
|       9 | Akis        | Petretzikis |
|       2 | Yiannis     | Lucacos     |
|      21 | Panagiotis  | Tzamalis    |
|      22 | Pavlos      | Happilos    |
|       1 | Lefteris    | Lazarou     |
|      29 | Giannis     | Chatzikas   |
|      49 | Levis       | Kodhima     |
|      42 | Charalambos | Kotsonis    |
+---------+-------------+-------------+
8 rows in set (0.00 sec)

+---------+----------+----------+
| chef_id | name     | surname  |
+---------+----------+----------+
|       1 | Lefteris | Lazarou  |
|       2 | Yiannis  | Lucacos  |
|      29 | Giannis  | Chatzikas |
+---------+----------+----------+
3 rows in set (0.02 sec)
```

11.3

Εκφώνηση:

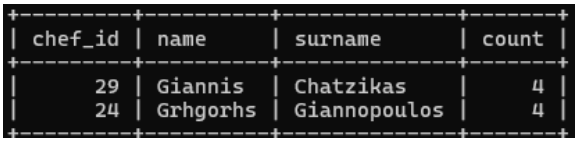
Βρείτε τους νέους μάγειρες (ηλικία < 30 ετών) που έχουν τις περισσότερες συνταγές.

Κώδικας:

```
WITH num_rec AS (
        SELECT MAX(count) as max_count
        FROM (
            SELECT COUNT(rc_id) as count
            FROM chef_recipe
            INNER JOIN chef ON chef.chef_id = chef_recipe.chef_id
            WHERE chef.age < 30
            GROUP BY chef.chef_id
        ) as rec_per_chef
    )
    SELECT chef.chef_id, chef.name, chef.surname, COUNT(rc_id) as count
    FROM chef_recipe
    INNER JOIN chef ON chef.chef_id = chef_recipe.chef_id
    WHERE chef.age < 30
    GROUP BY chef.chef_id, chef.name, chef.surname
    HAVING COUNT(rc_id) = (SELECT max_count FROM num_rec);
```

Listing 9: Query 3

Αποτελέσματα:



```
+---------+----------+--------------+-------+
| chef_id | name     | surname      | count |
+---------+----------+--------------+-------+
|      29 | Giannis  | Chatzikas    |     4 |
|      24 | Grhgorhs | Giannopoulos |     4 |
+---------+----------+--------------+-------+
```

11.4

Εκφώνηση:

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

Κώδικας:

```sql
SELECT chef_id, name , surname
     FROM chef
     WHERE chef_id NOT IN(
     SELECT first_judge_id FROM judges
     UNION
     SELECT second_judge_id FROM judges
     UNION
     SELECT third_judge_id FROM judges
     );
```

Listing 10: Query 4

Αποτελέσματα:

```
+---------+----------+--------------+
| chef_id | name     | surname      |
+---------+----------+--------------+
|      24 | Grhgorhs | Giannopoulos |
|      39 | Padelis  | Vouros       |
|      60 | Kostas   | Fretzios     |
+---------+----------+--------------+
3 rows in set (0.01 sec)
```

11.5

Εκφώνηση:

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

Κώδικας:

```sql
WITH appearances AS (
        SELECT season_id,judge_id, COUNT(episode_id) AS appearances_of_judge
    FROM (
                SELECT episode_id,season_id, first_judge_id as judge_id
                FROM judges
                UNION ALL
        SELECT episode_id,season_id, second_judge_id as judge_id
                FROM judges
        UNION ALL
        SELECT episode_id,season_id, third_judge_id as judge_id
                FROM judges) AS judges_combined
        GROUP BY season_id,judge_id
),
judge_appearances AS (
```

```sql
        SELECT a.season_id as season_id, a.judge_id as judge_id , c.name as name , c.surname as surname, a.appearances_of_judge
             as appearances_of_judges
        FROM appearances a
        INNER JOIN chef c ON a.judge_id = c.chef_id
        WHERE a.appearances_of_judge > 3
        GROUP BY season_id, judge_id)
SELECT j1.season_id, j1.judge_id as judge1_id, j1.name as judge1_name, j1.surname as judge1_surname,j2.judge_id as judge2_id, j2.
      name as judge2_name, j2.surname as judge2_surname, j1.appearances_of_judges
FROM judge_appearances j1
JOIN judge_appearances j2
ON j1.season_id = j2.season_id AND j1.judge_id < j2.judge_id
WHERE j1.appearances_of_judges = j2.appearances_of_judges ;
```

Listing 11: Query 5

Αποτελέσματα:

```
+-----------+-----------+-------------+----------------+-----------+-------------+----------------+-----------------------+
| season_id | judge1_id | judge1_name | judge1_surname | judge2_id | judge2_name | judge2_surname | appearances_of_judges |
+-----------+-----------+-------------+----------------+-----------+-------------+----------------+-----------------------+
|         2 |         8 | Giovanni    | Scaraggi       |        12 | Stavros     | Varthalitis    |                     4 |
+-----------+-----------+-------------+----------------+-----------+-------------+----------------+-----------------------+
1 row in set (0.00 sec)
```

11.6

Εκφώνηση:

Πολλές συνταγές καλύπτουν περισσότερες από μια ετικέτες. Ανάμεσα σε ζεύγη πεδίων (π.χ. brunch και κρύο πιάτο) που είναι κοινά στις συνταγές, βρείτε τα 3 κορυφαία (top-3) ζεύγη που εμφανίστηκαν σε επεισόδια6 Για το ερώτημα αυτό η απάντηση σας θα πρέπει να περιλαμβάνει εκτός από το ερώτημα (query), εναλλακτικό Query Plan (πχ με force index), τα αντίστοιχα traces και τα συμπεράσματα σας από την μελέτη αυτών.

Παραδοχές:

Στο ερώτημα αυτό παρατηρήσαμε ότι οι στήλες οι οποίες χρησιμοποιούνται στο ερώτημα έχουν ήδη ευρετύριο αφού είναι είτε πρωτεύον είτε δευτερεύον κλειδί στους αντίστοιχους πίνακες. Επομένως η επιβολή(force) ενός ευρετυρίου δεν θα είχε μεταβολές στην εκτέλεση του ερωτήματος. Για αυτό το λόγο το εναλλακτικό Query Plan απαγορεύει την χρήση των ευρετυρίων ώστε να παρατηρήσουμε τις διαφορές.

Κώδικας:

```sql
SELECT  a.lb_name AS label1,
        b.lb_name AS label2,
        COUNT(*) AS count
FROM episode_entries e
INNER JOIN  recipe_label a ON e.rc_id = a.rc_id
INNER JOIN  recipe_label b ON (a.lb_name < b.lb_name AND a.rc_id = b.rc_id)
GROUP BY label1, label2
ORDER BY count DESC
LIMIT 3;

/*Alternate Query Plan*/
SELECT  a.lb_name AS label1,
```

```
        b.lb_name AS label2,
        COUNT(*) AS count
FROM episode_entries e IGNORE INDEX(rc_id)
INNER JOIN  recipe_label a IGNORE INDEX(PRIMARY,lb_name) ON e.rc_id = a.rc_id
INNER JOIN  recipe_label b IGNORE INDEX(PRIMARY,lb_name) ON (a.lb_name < b.lb_name AND a.rc_id = b.rc_id)
GROUP BY label1, label2
ORDER BY count DESC
LIMIT 3;
```

Listing 12: Query 6

Αποτελέσματα:

Τα αποτελέσματα που πέρνουμε και από τα δύο query είναι τα εξής:

```
+-----------------+------------------+-------+
| label1          | label2           | count |
+-----------------+------------------+-------+
| Egg Free Diet   | Nuts Free Diet   |   122 |
| Egg Free Diet   | Main Course      |    79 |
| Nuts Free Diet  | Vegetarian Diet  |    71 |
+-----------------+------------------+-------+
3 rows in set (0.01 sec)
```

Όμως χρησιμοποιώντας τη δεσμευμένη λέξη Explain για τα queries μπορούμε να δούμε τα εξής :

```
+----+-------------+-------+------------+-------+---------------+---------+---------+-------------------------+------+----------+----------------------------------------------+
| id | select_type | table | partitions | type  | possible_keys | key     | key_len | ref                     | rows | filtered | Extra                                        |
+----+-------------+-------+------------+-------+---------------+---------+---------+-------------------------+------+----------+----------------------------------------------+
|  1 | SIMPLE      | a     | NULL       | index | PRIMARY,lb_name | lb_name | 122     | NULL                    |  162 |   100.00 | Using index; Using temporary; Using filesort |
|  1 | SIMPLE      | b     | NULL       | ref   | PRIMARY,lb_name | PRIMARY | 4       | cooking_competition.a.rc_id |    2 |    33.33 | Using where; Using index                     |
|  1 | SIMPLE      | e     | NULL       | ref   | rc_id         | rc_id   | 4       | cooking_competition.a.rc_id |    7 |   100.00 | Using index                                  |
+----+-------------+-------+------------+-------+---------------+---------+---------+-------------------------+------+----------+----------------------------------------------+
3 rows in set, 1 warning (0.00 sec)
```

```
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+----------------------------------------------------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra                                              |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+----------------------------------------------------+
|  1 | SIMPLE      | a     | NULL       | ALL  | NULL          | NULL | NULL    | NULL |  162 |   100.00 | Using temporary; Using filesort                    |
|  1 | SIMPLE      | b     | NULL       | ALL  | NULL          | NULL | NULL    | NULL |  162 |     0.62 | Using where; Using join buffer (hash join)         |
|  1 | SIMPLE      | e     | NULL       | ALL  | NULL          | NULL | NULL    | NULL |  500 |     1.49 | Using where; Using join buffer (hash join)         |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+----------------------------------------------------+
3 rows in set, 1 warning (0.00 sec)
```

Παρατηρούμε λοιπόν ότι με τη χρήση του ευρετυρίου ελέγχονται σημαντικά πιο λίγες εγγραφές των πινάκων κατα την εκτέλεση των JOIN σε αντίθεση με την περίπτωση που δεν χρησιμοποιούντα τα ευρετύρια όπου γίνεται scan στους πίνακες και άρα ελέγχονται όλες οι εγγραφές κάθε φορά.

11.7

Εκφώνηση:

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

Κώδικας:

```sql
WITH max_appearances AS(
        SELECT MAX(count_appearances) as max_count
    FROM(
                SELECT COUNT(chef_id) as count_appearances
                FROM episode_entries
        GROUP BY chef_id) AS chef_appearances
)
SELECT c.chef_id, c.name, c.surname, COUNT(e.chef_id) as appearances
FROM episode_entries e
INNER JOIN chef c ON e.chef_id = c.chef_id
GROUP BY chef_id
HAVING appearances < ((SELECT max_count FROM max_appearances) - 4);
```

Listing 13: Query 7

Αποτελέσματα:

chef_id	name	surname	appearances
1	Lefteris	Lazarou	6
2	Yiannis	Lucacos	4
3	Dimitris	Skarmoutsos	6
4	Sotiris	Kontizas	10
5	Panos	Ioannidis	11
6	Leonidas	Koutsopoulos	11
7	Margarita	Nikolaidi	8
8	Giovanni	Scaraggi	3
10	Christos	Glossidis	6
11	Giorgos	Porfiris	9
13	Stavros	Georgiou	9
14	Spiridoula	Karampoutaki	8
15	Ilias	Kaziolis	6
16	Maria	Bay	6
17	Ioannis	Charalampous	5
18	Dimitris	Bellos	7
19	John	Reano	11
20	Maria	Lazaridou	9
21	Panagiotis	Tzamalis	5
22	Pavlos	Happilos	3
23	Stefanos	Hilas	3
24	Grhgorhs	Giannopoulos	9
26	Panagiotis	Avgenikos	7
27	Vaggelis	Filaktakis	10
28	Dimitris	Politakis	5
30	Nikitas	Filippakis	7
31	Konstantinos	Kokkotas	8
35	Giorgos	Orfanos	5
36	Zahir	Yawary	7
37	Manos	Sarris	11
38	Panos	Togias	4
40	Stamatis	Kovaios	5
41	Tasos	Kiriklakis	11
42	Charalambos	Kotsonis	1
43	Xristos	Barkas	6
45	Gogo	Delogianni	8
46	Dimitris	Papapanagiotou	8
47	Zachos	Papadakis	11
48	Lambros	Vakiaros	11
50	Ektoras	Botrini	10
52	Athinagoras	Kostakos	7
53	Aris	Vezenes	5
54	Nikos	Thomas	11
55	Christina	Christophi	4
56	Ilias	Panagiotou	6
57	Lefteris	Zafeiropoulos	9
58	Marios	Ioannou	9
59	Christos	Moiras	9
60	Kostas	Fretzios	5

49 rows in set (0.01 sec)

11.8

Εκφώνηση:

Σε ποιο επεισόδιο χρησιμοποιήθηκαν τα περισσότερα εξαρτήματα (εξοπλισμός); Ομοίως με ερώτημα 3.6, η απάντηση σας θα πρέπει να περιλαμβάνει εκτός από το ερώτημα (query), εναλλακτικό Query Plan (πχ με force index), τα αντίστοιχα traces και τα συμπεράσματα σας από την μελέτη αυτών.

Παραδοχές:

Στο ερώτημα αυτό, όμοια με το ερώτημα 6 , παρατηρήσαμε ότι οι στήλες οι οποίες χρησιμοποιούνται στο ερώτημα έχουν ήδη ευρετύριο αφού είναι είτε πρωτεύον είτε δευτερεύον κλειδί στους αντίστοιχους πίνακες. Επομένως η επιβολή(force) ενός ευρετυρίου δεν θα είχε μεταβολές στην εκτέλεση του ερωτήματος. Για αυτό το λόγο το εναλλακτικό Query Plan απαγορεύει την χρήση των ευρετυρίων ώστε να παρατηρήσουμε τις διαφορές.

Κώδικας:

```sql
SELECT ep.episode_id, ep.season_id , COUNT(*) as number_of_eq
FROM episode_entries ep
INNER JOIN equipment_used eq
ON ep.rc_id = eq.rc_id
GROUP BY ep.episode_id,ep.season_id
ORDER BY number_of_eq DESC
LIMIT 1;

/*Alternate Query Plan*/
SELECT ep.episode_id, ep.season_id , COUNT(*) as number_of_eq
FROM episode_entries ep IGNORE INDEX(rc_id,episode_id)
INNER JOIN equipment_used eq IGNORE INDEX(PRIMARY,eq_id)
ON ep.rc_id = eq.rc_id
GROUP BY ep.episode_id,ep.season_id
ORDER BY number_of_eq DESC
LIMIT 1;
```

Listing 14: Query 8

Αποτελέσματα:

Τα αποτελέσματα που πέρνουμε και από τα δύο query είναι τα εξής:

```
+------------+-----------+--------------+
| episode_id | season_id | number_of_eq |
+------------+-----------+--------------+
|         10 |         5 |           39 |
+------------+-----------+--------------+
1 row in set (0.01 sec)
```

Όμως χρησιμοποιώντας τη δεσμευμένη λέξη Explain για τα queries μπορούμε να δούμε τα εξής :

```
+----+-------------+-------+------------+------+---------------+---------+---------+-------------------------------+------+----------+----------------------------------------------+
| id | select_type | table | partitions | type | possible_keys | key     | key_len | ref                           | rows | filtered | Extra                                        |
+----+-------------+-------+------------+------+---------------+---------+---------+-------------------------------+------+----------+----------------------------------------------+
|  1 | SIMPLE      | ep    | NULL       | ALL  | episode_id,rc_id | NULL    | NULL    | NULL                          |  500 |   100.00 | Using temporary; Using filesort              |
|  1 | SIMPLE      | eq    | NULL       | ref  | PRIMARY       | PRIMARY | 4       | cooking_competition.ep.rc_id |    2 |   100.00 | Using index                                  |
+----+-------------+-------+------------+------+---------------+---------+---------+-------------------------------+------+----------+----------------------------------------------+
2 rows in set, 1 warning (0.00 sec)

+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+----------------------------------------------------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra                                              |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+----------------------------------------------------+
|  1 | SIMPLE      | eq    | NULL       | ALL  | NULL          | NULL | NULL    | NULL |  190 |   100.00 | Using temporary; Using filesort                    |
|  1 | SIMPLE      | ep    | NULL       | ALL  | NULL          | NULL | NULL    | NULL |  500 |     1.49 | Using where; Using join buffer (hash join)         |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+----------------------------------------------------+
2 rows in set, 1 warning (0.00 sec)
```

Παρατηρούμε λοιπόν ότι με τη χρήση του ευρετυρίου ελέγχονται σημαντικά πιο λίγες εγγραφές των πινάκων κατα την εκτέλεση των JOIN σε αντίθεση με την περίπτωση που δεν χρησιμοποιούντα τα ευρετύρια όπου γίνεται scan στους πίνακες και άρα ελέγχονται όλες οι εγγραφές κάθε φορά.

11.9

Εκφώνηση:

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

Παραδοχές:

Στο ερώτημα αυτό κάνουμε την παραδοχή ότι ζητείται ο μέσος όρος γραμμαρίων υδατανθράκων ανά μερίδα.

Κώδικας:

```sql
SELECT e.season_id,AVG(r.carbs_per_portion) AS average_carbs_per_season
FROM episode_entries e
INNER JOIN recipe r ON e.rc_id = r.recipe_id
GROUP BY e.season_id;
```

Listing 15: Query 9

Αποτελέσματα:

```
+-----------+--------------------------+
| season_id | average_carbs_per_season |
+-----------+--------------------------+
|         1 |                20.804800 |
|         2 |                21.774400 |
|         3 |                24.084800 |
|         4 |                22.460300 |
|         5 |                24.573500 |
+-----------+--------------------------+
5 rows in set (0.01 sec)
```

11.10

Εκφώνηση:

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

Κώδικας:

```sql
WITH nt_appearances AS (
        SELECT nt_name, season_id, COUNT(episode_id) AS nt_app_per_season
        FROM episode_entries
    GROUP BY nt_name, season_id
    HAVING COUNT(episode_id) >= 3
),
consecutive_appearances AS (
    SELECT n1.nt_name, n1.nt_app_per_season + n2.nt_app_per_season AS consecutive_appearances, n1.season_id AS season_1, n2.
          season_id AS season_2
    FROM nt_appearances n1
    JOIN nt_appearances n2
    ON n1.nt_name = n2.nt_name AND n1.season_id = n2.season_id - 1
),
appearance_combinations AS (
    SELECT
    consecutive_appearances, season_1, COUNT(*) AS combination_count
```

```
    FROM consecutive_appearances
    GROUP BY consecutive_appearances, season_1
    HAVING COUNT(*) > 1
)
SELECT ca.nt_name, ca.consecutive_appearances, ca.season_1, ca.season_2
FROM consecutive_appearances ca
JOIN appearance_combinations ac
ON ca.consecutive_appearances = ac.consecutive_appearances AND ca.season_1 = ac.season_1
ORDER BY ca.season_1, ca.season_2, ca.consecutive_appearances;
```

Listing 16: Query 10

Αποτελέσματα:

nt_name	consecutive_appearances	season_1	season_2
South African	6	1	2
Arab	6	1	2
Cypriot	7	1	2
Italian	7	1	2
American	7	1	2
German	9	1	2
Chinese	9	1	2
Lebanese	9	1	2
Australian	10	1	2
Polish	10	1	2
Spanish	10	1	2
Brazilian	10	1	2
Korean	12	1	2
French	12	1	2
Indian	12	1	2
Egyptian	12	1	2
Chinese	7	2	3
South African	7	2	3
Lebanese	8	2	3
Greek	8	2	3
Mexican	9	2	3
British	9	2	3
American	10	2	3
Arab	10	2	3
Australian	10	2	3
Italian	10	2	3
Indian	10	2	3
Spanish	11	2	3
Brazilian	11	2	3
Polish	11	2	3
German	11	2	3
Russian	11	2	3
British	7	3	4
Russian	7	3	4
Chinese	7	3	4
Arab	10	3	4
Cypriot	10	3	4
German	10	3	4
American	11	3	4
Polish	11	3	4
Mexican	11	3	4
Japanese	11	3	4
Spanish	11	3	4
Greek	12	3	4
Italian	12	3	4
French	13	3	4
Brazilian	13	3	4
Arab	8	4	5
Russian	8	4	5
Egyptian	8	4	5
British	9	4	5
Indian	9	4	5
Japanese	9	4	5
South African	9	4	5
Brazilian	10	4	5
Spanish	10	4	5
American	10	4	5
Cypriot	11	4	5
Italian	11	4	5
Mexican	12	4	5
Polish	12	4	5
French	12	4	5
Greek	12	4	5

63 rows in set (0.01 sec)

11.11

Εκφώνηση:

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

Κώδικας:

```
WITH judge_score AS(
    (SELECT e.chef_id as chef_id, e.score1 as score, j.first_judge_id as judge_id
    FROM episode_entries e
    INNER JOIN judges j ON (e.episode_id = j.episode_id AND e.season_id = j.season_id))
    UNION ALL
    (SELECT e.chef_id as chef_id, e.score2 as score, j.second_judge_id as judge_id
    FROM episode_entries e
    INNER JOIN judges j ON (e.episode_id = j.episode_id AND e.season_id = j.season_id))
```

```
        UNION ALL
    (SELECT e.chef_id as chef_id, e.score3 as score, j.third_judge_id as judge_id
    FROM episode_entries e
    INNER JOIN judges j ON (e.episode_id = j.episode_id AND e.season_id = j.season_id))
)
SELECT chef_id, SUM(score) as total_score, judge_id
FROM judge_score
GROUP BY chef_id,judge_id
ORDER BY total_score DESC
LIMIT 5;
```

Listing 17: Query 11

Αποτελέσματα:

```
+---------+-------------+----------+
| chef_id | total_score | judge_id |
+---------+-------------+----------+
|      29 |          18 |        8 |
|      12 |          15 |       14 |
|      48 |          14 |        8 |
|      41 |          14 |       21 |
|      26 |          14 |       27 |
+---------+-------------+----------+
5 rows in set (0.01 sec)
```

11.12

Εκφώνηση:

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

Κώδικας:

```
 WITH RankedEpisodes AS (
        SELECT episode_entries.season_id, episode_entries.episode_id, SUM(recipe.difficulty) AS total_difficulty,
    ROW_NUMBER() OVER (PARTITION BY episode_entries.season_id ORDER BY SUM(recipe.difficulty)DESC) AS row_num
    FROM
    episode_entries INNER JOIN recipe
    ON episode_entries.rc_id=recipe.recipe_id
    GROUP BY
    episode_entries.season_id,episode_entries.episode_id
)
SELECT season_id,episode_id,total_difficulty
FROM RankedEpisodes
WHERE row_num=1
ORDER BY season_id;
```

Listing 18: Query 12

Αποτελέσματα:

```
+-----------+------------+------------------+
| season_id | episode_id | total_difficulty |
+-----------+------------+------------------+
|         1 |          6 |               31 |
|         2 |          8 |               32 |
|         3 |          7 |               28 |
|         4 |          2 |               30 |
|         5 |         10 |               29 |
+-----------+------------+------------------+
5 rows in set (0.01 sec)
```

11.13

Εκφώνηση:

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

Κώδικας:

```sql
SELECT e.episode_id, e.season_id,
     SUM(CASE
            WHEN c.professional_title = 'chef' THEN 1
            WHEN c.professional_title = 'sous chef' THEN 2
            WHEN c.professional_title = 'A chef' THEN 3
            WHEN c.professional_title = 'B chef' THEN 4
            WHEN c.professional_title = 'C chef' THEN 5
            ELSE 0
            END) AS total_prof_score
FROM chefs_in_episodes e
INNER JOIN
chef c ON e.chef_id=c.chef_id
GROUP BY e.episode_id,e.season_id
ORDER BY total_prof_score DESC
LIMIT 1;
```

Listing 19: Query 13

Αποτελέσματα:

```
+------------+-----------+------------------+
| episode_id | season_id | total_prof_score |
+------------+-----------+------------------+
|          5 |         2 |               47 |
+------------+-----------+------------------+
1 row in set (0.00 sec)
```

11.14

Εκφώνηση:

Ποια θεματική ενότητα έχει εμφανιστεί τις περισσότερες φορές στο διαγωνισμό;

Κώδικας:

```sql
SELECT ts_name, COUNT(*) AS count
FROM recipe_thematic_section INNER JOIN episode_entries
ON recipe_thematic_section.rc_id=episode_entries.rc_id
GROUP BY ts_name
ORDER BY count DESC
LIMIT 1;
```

Listing 20: Query 14

Αποτελέσματα:

```
+-----------------+-------+
| ts_name         | count |
+-----------------+-------+
| Sunday Suppers |   121 |
+-----------------+-------+
1 row in set (0.00 sec)
```

11.15

Εκφώνηση:

Ποιες ομάδες τροφίμων δεν έχουν εμφανιστεί ποτέ στον διαγωνισμό;

Παραδοχές:

Στο ερώτημα αυτό κάνουμε την παραδοχή ότι ζητούνται ομάδες τροφίμων που δεν έχουν εμφανιστεί σαν ομάδα τροφίμων κάποιου βασικού συστατικού μιας συνταγής σε όλο το διαγωνισμό

Κώδικας:

```sql
SELECT ing_g_name
FROM ingredient_group
WHERE ing_g_name NOT IN ( SELECT DISTINCT ig.ing_group
                                        FROM episode_entries e
                            INNER JOIN recipe r ON e.rc_id = r.recipe_id
                            INNER JOIN ingredient ig ON r.basic_ingredient = ig.ing_id);
```

Listing 21: Query 15

Αποτελέσματα:

```
+-------------------------+
| ing_g_name              |
+-------------------------+
| Fats/oils               |
| Non-alcoholic beverages |
| Alcoholic beverages     |
+-------------------------+
3 rows in set (0.01 sec)
```

12. Οδηγίες Εγκατάστασης Βάσης

- Το github repository της βάσης δεδομένων είναι το εξής :
  https://github.com/kallishis/Databases.git
- Εγκαθιστούμε αρχικά τα git bash, node js, mySQL, mySQL Workbench
- Έπειτα μέσω του git bash μπορούμε να κάνουμε clone το repository με την εξής εντολή:
  git clone https://github.com/kallishis/Databases.git
- Στο MySQL Workbench τρέχουμε τα αρχεία DDL.sql, DML.sql, roles.sql (Με αυτή τη σειρά!)
- Έπειτα σε ένα editor (Εμείς χρησιμοποιήσαμε το Visual Code Studio) εγκαθιστούμε την mysql2 μέσω του terminal με την εντολή :
  npm install mysql2
- Στη συνέχεια μεταβαίνουμε στο φάκελο όπου έχουμε αποθηκεύσει τα αρχεία που αντιγράψαμε από το repository και τρέχουμε τα αρχεία random.js και random_judges.js με τις εντολές
  node random.js
  node random_judges.js
- Μετά και από το τελευταίο στάδιο μπορούμε να συνδεθούμε στη βάση και να εκτελέσουμε queries.