ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΡΓΑΣΤΗΡΙΟ ΣΥΣΤΗΜΑΤΩΝ ΒΑΣΕΩΝ ΓΝΩΣΕΩΝ ΚΑΙ ΔΕΔΟΜΕΝΩΝ

ΕΞΑΜΗΝΙΑΙΑ ΕΡΓΑΣΙΑ

Ακ. έτος 2023-2024, 6ο εξάμηνο, ΣΗΜΜΥ

Ειρήνη Σιμιτζή 03121063
Οικονόμου Γεώργιος 03121103
Αικατερίνη Ρουσουνέλου 03121846

Απαίτηση 1 : Διάγραμμα ER

Στόχος της συγκεκριμένης εργασίας ήταν ο σχεδιασμός και η υλοποίηση μίας βάσης δεδομένων, που απαιτείται για έναν διαγωνισμό μαγειρικής. Πρωταρχικό μας βήμα ήταν η ευκρινής διαμέριση των οντοτήτων (Entity Sets) της βάσης καθώς βέβαια, και των σχέσεων που τις συνδέουν, προκειμένου να εξασφαλίσουμε μία -όσο το δυνατόν- σαφέστερη προσέγγιση. Έτσι, χρησιμοποιήσαμε το μοντέλο οντοτήτων - συσχετίσεων, γνωστό ως ER Model. Επεξεργαστήκαμε το κείμενο της εκφώνησης της άσκησης, αντλώντας τα δεδομένα και προκύπτουν τα συγκεκριμένα σύνολα (Entity Sets) (τα παρουσιάζουμε έτσι όπως φαίνονται και στο διάγραμμα):

01. Recipe	06. Steps	11. Cuisine
02. MealType	07. Equipment	12. Episode
03. Label	08. NutritionalInfo	13. Cook
04. Ingredients	09. Tip	14. Specialization
05. FoodGroup	10. Theme	15. Score

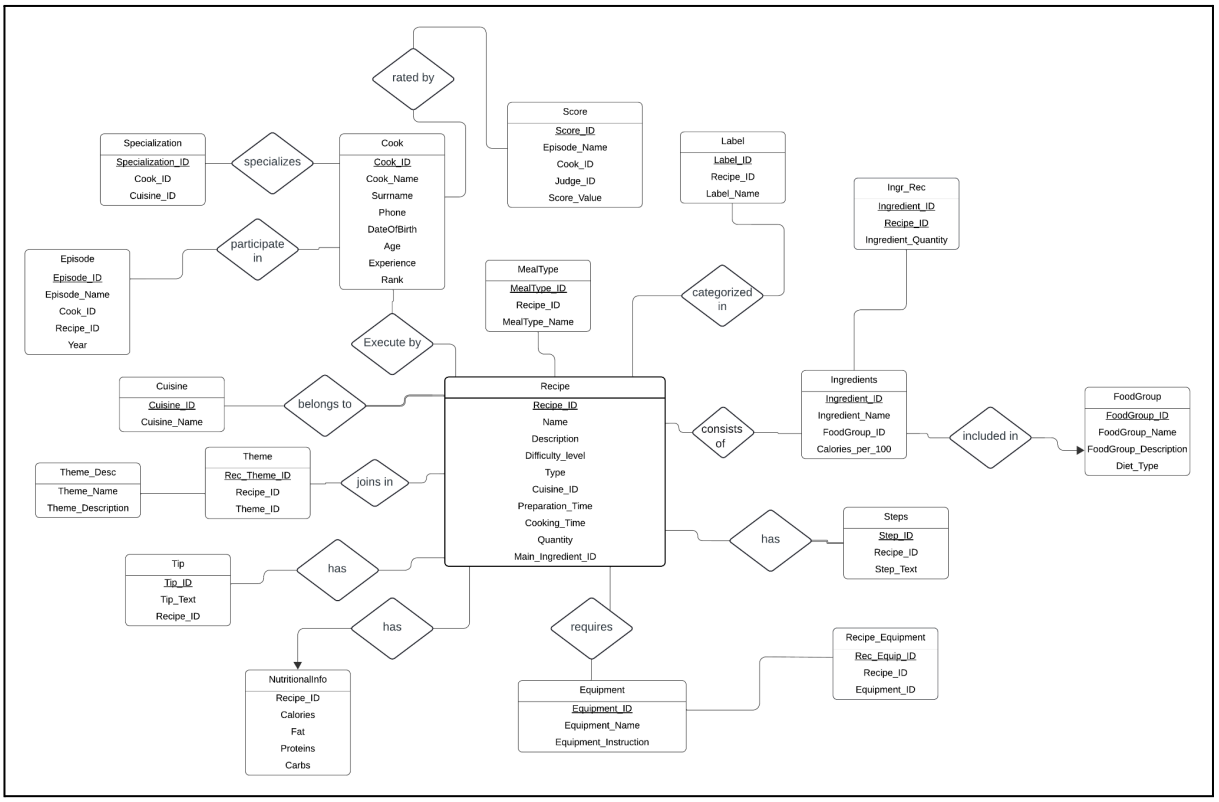
Τα παραπάνω συμβολίζουν τις συνταγές (1), τον τύπο αυτών (2), την ετικέτα τους (3) (brunch, πρωινό κλπ) και την θεματική κατηγορία theme (10) τους. Ακολουθούν τα υλικά (4) και η συγκεκριμένη κατηγορία στην οποία κάθε υλικό ανήκει (4,5). Εν συνεχεία, έχουμε τα σειριακά βήματα που χαρακτηρίζουν την συνταγή (6), τον απαιτούμενο εξοπλισμό (7), καθώς και διάφορες συμβουλές για κάθε συνταγή (9). Επιπρόσθετα, έχουμε τις διατροφικές πληροφορίες για τα μακροθρεπτικά της συνταγής (8) και πληροφορίες σχετικά με εθνική κουζίνα (11). Τέλος, έχουμε συμπεριλάβει την βασική οντότητα των μαγείρων (13), την ειδικότητα τους (14) και πληροφορίες αναφορικά με τα επεισόδια του διαγωνισμού (14) και τον βαθμό κάθε διαγωνιζόμενου. Όλες αυτές οι οντότητες διέπονται από αλυσιδωτές σχέσεις που τις συνδέουν, φανερώνοντας το είδος της "εξάρτησης" που έχει η μία από την άλλη. Συγκεκριμένα:

execute by: recipe - cook	described by: recipe - nutritionalinfo	includes in: ingredients - foodgroup

belongs to: recipe - cuisine	requires: recipe - equipment	rated by: cook - score
joins in: recipe - theme	consists of: recipe - ingredients	specializes: cook - specialization
has: recipe - tip, recipe - steps	categorized in: recipe - label, recipe - mealtype	participate in: cook - episode

Ουσιαστικά, στην παραπάνω λίστα παρουσιάσαμε τις σχέσεις μεταξύ των οντοτήτων οι οποίες ολοκληρώνουν την προσέγγιση μας. Ειδικότερα, έχουμε σχέση μεταξύ μαγείρων και συνταγών η οποία δηλώνει το ποιος μάγειρας εκτελεί ποια συνταγή. Ακολούθως, δημιουργήσαμε την belongs_to για να εκφράσουμε σε ποια κουζίνα ανήκει η κάθε συνταγή και την joins για να δηλώσουμε το σε ποια θεματική κατηγορία ανήκει η τελευταία. Εν συνεχεία, συνδέσαμε συνταγές με tips και steps μέσω της σχέσης has για να έχουμε πρόσβαση μέσω της συνταγής στις συμβουλές για αυτή και τα βήματα εκτέλεσης της. Συνεχίζοντας με το core entity των recipes έχουμε την requires και την consists μέσω των οποίων αντιλαμβανόμαστε τον απαιτούμενο gear και τα απαραίτητα συστατικά κάθε συνταγής. Επιπρόσθετα, έχουμε συσχέτιση μεταξύ συνταγών και ετικετών και meal_type για να έχουμε πρόσβαση στα τελευταία μέσω καθεμιάς συνταγής. Όσον αφορά τα συστατικά - υλικά τα συνδέουμε με το food_group για να έχουμε το σε ποια κατηγορία ανήκει κάθε υλικό. Τέλος, αφήσαμε τα δρώμενα του διαγωνισμού. Συνδέσαμε τον μάγειρα με τον τομέα στον οποίο ειδικεύεται και για την υλοποίηση του contest φτιάξαμε δυο σχέσεις αναφορικά με το ποιος μάγειρας συμμετέχει σε ποιο επεισόδιο (participate in) και άλλη μια για το ποιος μάγειρας βαθμολογεί τον υποψήφιο του διαγωνισμού (graded_by).

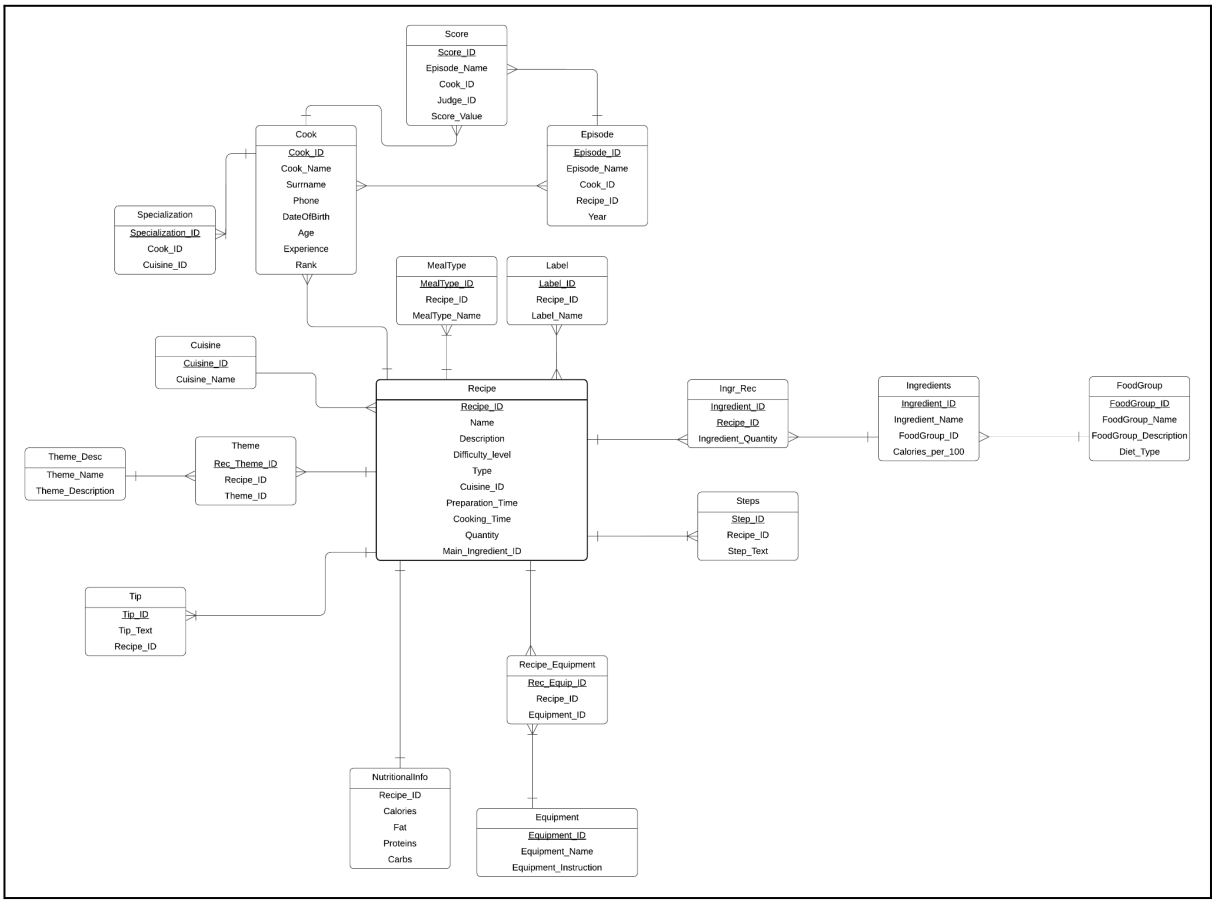
Έτσι, το ER Diagram θα έχει την παρακάτω μορφή. Να σημειωθεί στο σημείο αυτό ότι με υπογραμμισμένα γράμματα απεικονίζονται τα Primary Keys του κάθε Entity. Ακόμη, όπως εύκολα αντιλαμβανόμαστε τα χαρακτηριστικά ορισμένων οντοτήτων μπορούν να πάρουν πάνω από μία τιμή (όπως τα τηλέφωνα των μαγείρων, τα tips κάθε συνταγής κλπ).



Απαίτηση 2 : Σχεσιακό Μοντέλο

Επειδή η γλώσσα που θα χρησιμοποιηθεί για την κατασκευή της βάσης είναι η SQL, απαιτούμε ένα σύνολο πινάκων. Απαιτούμε πρακτικά την μετατροπή του προηγούμενος διαγράμματος σε ένα σχεσιακό διάγραμμα, το οποίο παρουσιάζει τον τρόπο κατηγοριοποίησης των δεδομένων από την ανθρώπινη οπτική πλευρά. Η διαδικασία αυτής της μετατροπής ξεκινάει με "μεταμόρφωση" των οντοτήτων σε πίνακες (οι οποίοι θα αντιπροσωπεύουν τις σχέσεις), ενώ παράλληλα η αφαιρετική μέθοδος είναι ιδιαίτερα χρήσιμη.

Σχετικά με τους περιορισμούς, όλα τα πεδία δεν επιτρέπεται να είναι NULL και για κάθε σχέση συνδυάσαμε το ER με το σχεσιακό λαμβάνοντας υπόψιν και τα foreign keys, στα σημεία όπου υπάρχουν. Οι περιορισμοί εξετάζονται αναλυτικά στο Schema. Λόγω του ότι δεν βρήκαμε κάποιο index όπου να μας βελτιώνει την τρέχουσα κατάσταση, δεν ορίσαμε περαιτέρω ευρετήρια για κάποιο από τα optimization στα queries. Έχουμε "υιοθετήσει" τα ευρετήρια που είναι by default ορισμένα στο περιβάλλον του MySQL Workbench. Ουσιαστικά, κάθε primary key για κάθε entity και κάθε foreign key έχει από ένα ευρετήριο.



Απαίτηση 3: DDL και DML Scripts

- **DDL Script**

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL';


DROP SCHEMA IF EXISTS Cooking_Competition;
CREATE SCHEMA Cooking_Competition;
USE Cooking_Competition;

CREATE TABLE Recipe (
  Recipe_ID INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `Name`  VARCHAR(45) NOT NULL,
  `Type` ENUM ('cooking','pastry') NOT NULL,
   Cuisine_ID INT UNSIGNED,
  `Description` text NOT NULL,
  `Difficulty_level` TINYINT UNSIGNED NOT NULL CHECK (Difficulty_level BETWEEN 1 AND 5),
  Preparation_Time TIME,
  Cooking_Time TIME,
  Quantity INT UNSIGNED NOT NULL,
  Main_Ingredient_ID INT UNSIGNED,
  PRIMARY KEY  (Recipe_ID),
  CONSTRAINT `fk_recipe_cuisine` FOREIGN KEY (Cuisine_ID) REFERENCES  Cuisine(Cuisine_ID) ON
DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT `fk_recipe_ingredient` FOREIGN KEY (Main_Ingredient_ID) REFERENCES
Ingredients(Ingredients_ID) ON DELETE RESTRICT ON UPDATE CASCADE

);

CREATE TABLE Cuisine (
  Cuisine_ID INT UNSIGNED NOT NULL AUTO_INCREMENT,
  Cuisine_Name VARCHAR(50) NOT NULL,
  PRIMARY KEY  (Cuisine_ID)
);

CREATE TABLE MealType (
    MealType_ID INT UNSIGNED NOT NULL AUTO_INCREMENT,
    Recipe_ID INT UNSIGNED NOT NULL,
    MealType_Name VARCHAR(50) NOT NULL,

    PRIMARY KEY (MealType_ID),
    CONSTRAINT fk_recipe_mealtype FOREIGN KEY (Recipe_ID) REFERENCES Recipe(Recipe_ID) ON
DELETE RESTRICT ON UPDATE CASCADE
);
```

```
CREATE TABLE Label (
  Label_ID INT UNSIGNED NOT NULL AUTO_INCREMENT,
  Label_Name VARCHAR(50) NOT NULL,
  Recipe_ID INT UNSIGNED NOT NULL,
  PRIMARY KEY (Label_ID),
  CONSTRAINT `fk_recipe_label` FOREIGN KEY (Recipe_ID) REFERENCES Recipe(Recipe_ID) ON
DELETE RESTRICT ON UPDATE CASCADE
);


CREATE TABLE Equipment (
  Equipment_ID INT UNSIGNED NOT NULL AUTO_INCREMENT,
  Equipment_Name VARCHAR(50) NOT NULL,
  Equipment_Instruction TEXT NOT NULL,
  PRIMARY KEY (Equipment_ID)
);

CREATE TABLE Recipe_Equipment (
  Rec_Equip_ID INT UNSIGNED NOT NULL auto_increment,
  Recipe_ID INT UNSIGNED NOT NULL,
  Equipment_ID INT UNSIGNED NOT NULL,
  PRIMARY KEY (Rec_Equip_ID),
  CONSTRAINT `fk_recipe_equipment_recipe` FOREIGN KEY (Recipe_ID) REFERENCES
Recipe(Recipe_ID) ON DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT `fk_recipe_equipment_equipment` FOREIGN KEY (Equipment_ID) REFERENCES
Equipment(Equipment_ID) ON DELETE RESTRICT ON UPDATE CASCADE
);

CREATE TABLE Steps (
    Step_ID INT UNSIGNED NOT NULL AUTO_INCREMENT,
    Recipe_ID INT UNSIGNED NOT NULL,
    Step_Text TEXT NOT NULL,

    PRIMARY KEY (`Step_ID`),
    CONSTRAINT `fk_steps_recipe` FOREIGN KEY (`Recipe_ID`) REFERENCES `Recipe`(`Recipe_ID`) ON
DELETE RESTRICT ON UPDATE CASCADE
);

CREATE TABLE Ingredients (
    Ingredients_ID INT UNSIGNED AUTO_INCREMENT,
    Ingredient_Name VARCHAR(50) NOT NULL,
    FoodGroup_ID INT UNSIGNED,
    Calories_per_100 INT UNSIGNED,
    PRIMARY KEY (Ingredients_ID),
    CONSTRAINT `fk_ingredients_group` FOREIGN KEY(FoodGroup_ID) REFERENCES
FoodGroup(FoodGroup_ID) ON DELETE RESTRICT ON UPDATE CASCADE
);

CREATE TABLE Ingr_Rec (
   Recipe_ID INT UNSIGNED NOT NULL,
```

```
  Ingredients_ID  INT UNSIGNED NOT NULL,
  Ingredient_Quantity  DECIMAL(10, 2) NOT NULL,

 PRIMARY KEY (Recipe_ID , Ingredients_ID),
 CONSTRAINT `fk_recipe_ingredients_recipe` FOREIGN KEY (Recipe_ID) REFERENCES
Recipe(Recipe_ID) ON DELETE RESTRICT ON UPDATE CASCADE,
 CONSTRAINT `fk_recipe_ingredients_ingredient` FOREIGN KEY (Ingredients_ID) REFERENCES
Ingredients(Ingredients_ID) ON DELETE RESTRICT ON UPDATE CASCADE
);

CREATE TABLE FoodGroup (
   FoodGroup_ID INT UNSIGNED NOT NULL AUTO_INCREMENT,
   FoodGroup_Name VARCHAR(50) NOT NULL,
   FoodGroup_Description TEXT,
   Diet_Type VARCHAR(50) NOT NULL,
   PRIMARY KEY (FoodGroup_ID)
);

CREATE TABLE Cook (
Cook_ID INT UNSIGNED AUTO_INCREMENT,
Cook_Name VARCHAR(50),
Surname VARCHAR(50),
Phone VARCHAR(20),
DateOfBirth DATE,
Age INT,
Experience INT,
`Rank` ENUM("C' Cook", "B' Cook", "A' Cook","Sous-Chef","Head Chef"),

PRIMARY KEY (Cook_ID)
);

CREATE TABLE Specialization (
Specialization_ID INT UNSIGNED AUTO_INCREMENT,
Cook_ID INT UNSIGNED ,
Cuisine_ID INT UNSIGNED NOT NULL,

PRIMARY KEY (Specialization_ID),
CONSTRAINT `fk_Specialization_cuisine` FOREIGN KEY (Cuisine_ID) REFERENCES
Cuisine(Cuisine_ID) ON DELETE RESTRICT ON UPDATE CASCADE,
CONSTRAINT `fk_Specialization_Cook` FOREIGN KEY (Cook_ID) REFERENCES  Cook(Cook_ID) ON
DELETE RESTRICT ON UPDATE CASCADE
);

CREATE TABLE Theme (
Theme_Name VARCHAR(50) NOT NULL,
Theme_Description TEXT,
Theme_ID INT UNSIGNED NOT NULL auto_increment,
PRIMARY KEY (Theme_ID)
);
```

```
CREATE TABLE Rec_Them (
Rec_Them_ID INT UNSIGNED NOT NULL auto_increment,
Recipe_ID INT UNSIGNED NOT NULL,
Theme_ID INT UNSIGNED NOT NULL,
PRIMARY KEY (Rec_Them_ID),

CONSTRAINT `fk_theme_recipe` FOREIGN KEY (Recipe_ID) REFERENCES Recipe(Recipe_ID) ON
DELETE RESTRICT ON UPDATE CASCADE,
CONSTRAINT `fk_rec_them_theme` FOREIGN KEY (Theme_ID) REFERENCES Theme(Theme_ID) ON
DELETE RESTRICT ON UPDATE CASCADE
);



CREATE TABLE NutritionalInfo (
Recipe_ID INT UNSIGNED NOT NULL,
Calories INT UNSIGNED,
Fat INT UNSIGNED,
Proteins INT UNSIGNED,
Carbs INT UNSIGNED,
PRIMARY KEY (Recipe_ID),
  CONSTRAINT `fk_nutrinfo_recipe` FOREIGN KEY (Recipe_ID) REFERENCES Recipe(Recipe_ID) ON
DELETE RESTRICT ON UPDATE CASCADE

);

CREATE TABLE Tip (
Tip_ID INT UNSIGNED NOT NULL AUTO_INCREMENT ,
Tip_Text TEXT,
Recipe_ID INT UNSIGNED NOT NULL,

PRIMARY KEY (Tip_ID),
CONSTRAINT `fk_tip_recipe` FOREIGN KEY (Recipe_ID) REFERENCES Recipe(Recipe_ID) ON DELETE
RESTRICT ON UPDATE CASCADE

);

CREATE TABLE Score (
Score_ID INT UNSIGNED AUTO_INCREMENT,
Episode_Name INT,
Judge_ID INT UNSIGNED,
Cook_ID INT UNSIGNED,
Score_Value INT,
PRIMARY KEY (Score_ID)
);

CREATE TABLE Episode(
Episode_ID INT UNSIGNED AUTO_INCREMENT,
Episode_Name INT,
Cook_ID INT UNSIGNED,
```

```
Recipe_ID INT UNSIGNED,
`Year` INT ,
PRIMARY KEY (Episode_ID)
);
```

- **DML Script**

https://docs.google.com/document/d/1MX1FrbJyuOKbHGSipYM3HaLh5oPJ6rd-JQ9iDmTEV80/edit?usp=sharing

Απαίτηση 4: Queries

4.1 : Μέσος Όρος Αξιολογήσεων (σκορ) ανά μάγειρα καιΕθνική κουζίνα.

Παραθέτουμε πρώτα το τμήμα του κώδικα και στην συνέχεια την επεξήγησή του.

```
SELECT COOK_ID, AVR(Score_Value) AS Mean_Score
FROM Score
GROUP BY Cook_ID;
SELECT
        r.Cuisine_ID,
        c.Cuisine_Name,
        AVR(s.Score_Value) AS Mean_Score
FROM
        Score s
JOIN
        Episode.e ON s.Episode.Name = e.Episode_Name AND
s.Cook_ID=e.Cook_ID  JOIN
        Recipe r ON e.Recipe_ID=r.Recipe_ID
JOIN
        Cuisine c ON r.Cuisine_ID=c.Cuisine_ID
GROUP BY
        r.Cuisine_ID,
        c.Cuisine_Name;
```

Επεξήγηση: Εξηγώντας μία μία τις εντολές έχουμε:
SELECT Cook_ID : Αυτό επιλέγει το χαρακτηριστικό Cook_ID από τον πίνακα βαθμολογίας.
AVR(Score_Value) AS Mean_Score : Αυτό υπολογίζει τη μέση (μέση) τιμή του χαρακτηριστικού Score_Value για κάθε ομάδα Cook_ID και τη μετονομάζει σε Mean_Score.
FROM Score : Αυτό καθορίζει τον πίνακα από τον οποίο θα ανακτηθούν δεδομένα, σε αυτήν την περίπτωση, ο πίνακας βαθμολογίας.
GROUP BY Cook_ID; : Αυτό ομαδοποιεί τα αποτελέσματα κατά Cook_ID, επομένως η συνάρτηση AVG εφαρμόζεται σε κάθε ομάδα ξεχωριστά, δίνοντάς σας τη μέση βαθμολογία για κάθε μάγειρα. Για το μέσο score ανά εθνική κουζίνα, πρέπει να ομαδοποιήσουμε με βάση την εθνικότητα της κουζίνας και να κάνουμε join 4 relations.

Όταν χρησιμοποιείται ο όρος GROUP BY, συναρτήσεις συγκεντρωτικών στοιχείων όπως η AVG() λειτουργούν στο σύνολο των γραμμών σε κάθε ομάδα και όχι σε ολόκληρο τον πίνακα. Έτσι, για κάθε ξεχωριστό Cook_ID, το ερώτημα υπολογίζει τον μέσο όρο του Score_Value για όλες τις σειρές (βαθμολογίες) που έχουν αυτό το Cook_ID.

4.2 : Για δεδομένη Εθνική κουζίνα και έτος, ποιοι μάγειρες ανήκουν σε αυτή και ποιοι μάγειρες συμμετείχαν σε επεισόδια;

Το υποερώτημα αυτό έχει δύο ξεχωριστά μέρη επομένως :

- Μάγειρες που ανήκουν σε μια δεδομένη εθνική κουζίνα:

```
SELECT
        s.Cook_ID,
        c.Cook_Name,
        c.Surname
FROM
        Specialization s
JOIN
        Cook c ON s.Cook_ID=c.Cook_ID
JOIN
        Cuisine cu ON s.Cuisine_ID=cu.Cuisine_ID
WHERE
        cu.Cuisine_Name='National Cuisine Name';
```

Αυτό το ερώτημα SQL ανακτά πληροφορίες σχετικά με μάγειρες που ειδικεύονται σε μια συγκεκριμένη εθνική κουζίνα. Επιλέγει το Αναγνωριστικό του Μάγειρα, το Όνομα του Μάγειρα και το Επώνυμό του από τον πίνακα των Μαγείρων, συνδέοντάς τον με τον πίνακα Εξειδίκευσης μέσω του Αναγνωριστικού
του Μάγειρα. Επιπλέον, συνδέει τον πίνακα της Κουζίνας με τον πίνακα Εξειδίκευσης μέσω του Αναγνωριστικού της Κουζίνας για να ταιριάξει την εξειδίκευση με τον τύπο κουζίνας. Η δήλωση WHERE φιλτράρει τα αποτελέσματα για να περιλαμβάνει μόνο μάγειρες που ειδικεύονται στην καθορισμένη εθνική κουζίνα.

- Μάγειρες που συμμετείχαν σε επεισόδια κατά τη διάρκεια μιας δεδομένης χρονιάς:

```
SELECT DISTINCT
        e.Cook_ID,
        c.Cook_Name,
        c.Surname
FROM
        Episode e
JOIN
        Cook c ON e.Cook_ID=c.Cook_ID
WHERE
        e.Year=2024; -- Replace 2024 with the desired year
        AND e.Recipe_ID IS NOT NULL;
```

Αυτό το υπερώτημα επιστρέφει μοναδικές πληροφορίες σχετικά με μάγειρες που έχουν εμφανιστεί σε επεισόδια μαγειρικής την συγκεκριμένη χρονιά. Επιλέγει το Αναγνωριστικό του Μάγειρα, το Όνομα του Μάγειρα και το Επώνυμό του από τον πίνακα των Μαγείρων, συνδεόμενος με τον πίνακα των Επεισοδίων μέσω του Αναγνωριστικού του Μάγειρα. Η δήλωση WHERE φιλτράρει τα αποτελέσματα για να περιλάβουν μόνο επεισόδια που πραγματοποιήθηκαν το συγκεκριμένο έτος και έχουν καθορισμένο Recipe_ID.

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

```
SELECT
        c.Cook_ID,
        c.Cook_Name,
        c.Surname
        COUNT(DISTINCT e.Recipe_ID) AS Recipe_Count
FROM
        Cook c
JOIN
```

```
        Episode e ON c.Cook_ID=e.Cook_ID
WHERE
        c.Age < 30
        AND e.Recipe_ID IS NOT NULL
GROUP BY
        c.Cook_ID,
        c.Cook_Name,
        c.Surname
ORDER BY
        Recipe_Count DESC
LIMIT 5;
```

Εδώ επιστρέφουμε πληροφορίες σχετικά με μάγειρες που είναι κάτω των 30 ετών και έχουν μαγειρέψει τις περισσότερες συνταγές στα επεισόδια σε όλα τα χρόνια του διαγωνισμού. Επιλέγει το Αναγνωριστικό του Μάγειρα, το Όνομα του Μάγειρα και το Επώνυμό του από τον πίνακα των Μαγείρων. Χρησιμοποιεί τον πίνακα των Επεισοδίων για να συνδέσει τον μάγειρα με τα επεισόδια μέσω του Αναγνωριστικού του Μάγειρα, ενώ το Recipe_ID πρέπει να είναι μη μηδενικό. Το ερώτημα ομαδοποιεί τα αποτελέσματα ανά μάγειρα και ταξινομεί τους μάγειρες με βάση τον αριθμό των μοναδικών συνταγμάτων σε φθίνουσα σειρά. Τέλος, επιστρέφονται τα πρώτα 5 αποτελέσματα.

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

```
SELECT
        c.Cook_ID,
        c.Cook_Name,
        c.Surname
FROM
        Cook c
JOIN
        Episode e ON c.Cook_ID = e.Cook_ID
WHERE
        e.Recipe_ID IS NOT NULL
        AND c.Cook_ID NOT IN (
                SELECT
                        Cook_ID
                FROM
                        Episode
                WHERE
                        Recipe_ID IS NULL
        )
GROUP BY
        c.Cook_ID,
        c.Cook_Name,
        c.Surname;
```

Μέσω ενός join μεταξύ επεισοδίων και cooks επιλέγουμε μόνο αυτούς που δεν είναι συμμετέχοντες στον διαγωνισμό ώς κριτές. Τέλος, ομαδοποιούμε τα αποτελέσματα με βάση την ταυτότητα του μάγειρα, το όνομα και το επώνυμο του. Χρησιμοποιώντας το NOT IN (...), αποκλείουμε τυχόν μάγειρες που είναι παρόντες στη λίστα των κριτών, διασφαλίζοντας ότι στο τελικό αποτέλεσμα περιλαμβάνονται μόνο μάγειρες

που δεν έχουν υπηρετήσει ποτέ ως κριτές.

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

Η συνάρτηση GROUP_CONCAT(Judge_ID) συνενώνει τα Judge_IDs που έχουν το ίδιο Episode_Count και έτος.

```
WITH JudgeApperances AS (
        SELECT
                e.Cook_ID AS Judge_ID,
                e.Year,
                COUNT (*) AS Episode_Count
        FROM
                Episode e
        WHERE
                e.Recipe_ID IS NULL
        GROUP BY
                e.Cook_ID,
                e.Year
        HAVING
                COUNT(*) > 3
) ,
 JudgeApperancesGrouped AS (
        SELECT
                Year,
                Episode_Count,
                GROUP_CONCAT (Judge_ID) AS Judges
        FROM
                JudgeApperances
        GROUP BY
                Year, Episode_Count
        HAVING
                COUNT(Judge_ID) > 1
)
 SELECT
        Year,
        Episode_Count,
        Judges
 FROM
        JudgeAppearancesGrouped;
```

Προσδιορίσαμε τους κριτές επιλέγοντας σειρές στον πίνακα επεισοδίων όπου το Recipe_ID είναι NULL. Έπειτα μετρήσαμε τον αριθμό των επεισοδίων στα οποία εμφανίστηκε κάθε κριτής για
κάθε χρόνο και φιλτράραμε τα αποτελέσματα για να συμπεριλάβουμε μόνο εκείνους τους κριτές που έχουν περισσότερες από 3 εμφανίσεις.

4.6 : 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 επεισόδια. 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.

```
WITH LabelPairs AS (
    SELECT
        l1.Label_Name AS Label1,
        l2.Label_Name AS Label2,
        COUNT(*) AS Pair_Count
    FROM
        Label l1
    JOIN
        Label l2 ON l1.Recipe_ID=l2.Recipe_ID
            AND l1.Label_ID < l2.Label_ID - - Ensure distinct pairs
    JOIN
        Episode e ON l1.Recipe_ID = e.Recipe_ID
    GROUP BY
        l1.Ladel_Name,
        l2.Label_Name
) ,
TopPairs AS (
    SELECT
        Label1,
        Label2,
        Pair_Count,
       RANK() OVER (ORDER BY Pair_Coount DESC) AS pair_rank
    FROM
        LabelPairs
)
SELECT
    Label1,
    Label2,
    Pair_Count
FROM
    TopPairs
WHERE
    pair_rank <= 3;
```

Αυτό το ερώτημα SQL χρησιμοποιεί κοινές εκφράσεις πινάκων (CTEs) για να εντοπίσει και να κατατάξει τα πιο συνηθισμένα ζεύγη ετικετών που συνδέονται με συνταγές σε επεισόδια. Η πρώτη κοινή έκφραση πίνακα, "LabelPairs," επιλέγει τα ζεύγη ετικετών από τον πίνακα Label συνδέοντάς τον με τον εαυτό του βάσει του Recipe_ID και εξασφαλίζοντας ότι κάθε ζεύγος είναι μοναδικό. Επίσης, συνδέει τον πίνακα Episode για να φιλτράρει τα επεισόδια που σχετίζονται με αυτές τις ετικέτες. Το αποτέλεσμα περιλαμβάνει τα ονόματα των δύο ετικετών στο ζεύγος και τον αριθμό εμφανίσεων. Η δεύτερη κοινή έκφραση πίνακα, "TopPairs," κατατάσσει τα ζεύγη με βάση τον αριθμό εμφανίσεων χρησιμοποιώντας τη συνάρτηση RANK() και ταξινομημένα κατά φθίνουσα σειρά. Τέλος, το κύριο ερώτημα επιλέγει τα ονόματα των ετικετών και τον αριθμό εμφανίσεων από την κοινή έκφραση "TopPairs",

φιλτράροντας μόνο τα τρία κορυφαία ζεύγη με βάση την κατάταξή τους.

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

```
WITH CookAppearances AS (
	SELECT
		e.Cook_ID,
		COUNT(e.Episode_ID) AS Appearance_Count
	FROM
		Episode e
	WHERE
		e.Recipe_ID IS NOT NULL
	GROUP BY
		e.Cook_ID
) ,
MaxAppearances AS (
	SELECT
		MAX(Appearance_Count) AS Max_Appearance_Count
	FROM
		CookAppearances
)
SELECT
	c.Cook_ID,
	c.Cook_Name,
	c.Surname,
	ca.Appearance_Count
FROM
	Cook c
JOIN
	CookAppearances ca ON c.Cook_ID = ca.Cook_ID
JOIN
	MaxAppearances ma ON ca.Appearance_Count <= ma.Max_Appearance_Count - 5;
```

Το query που υλοποιήσαμε υπολογίζει τον αριθμό των εμφανίσεων για κάθε μάγειρα σε επεισόδια όπου υπάρχει μια συνταγή και εντοπίζει τους μάγειρες οι οποίοι βρίσκονται μέσα στους πέντε με τις περισσότερες εμφανίσεις. Η πρώτη κοινή έκφραση πίνακα, "CookAppearances," υπολογίζει τον αριθμό των εμφανίσεων για κάθε μάγειρα σε επεισόδια όπου ο Recipe_ID δεν είναι NULL. Η δεύτερη κοινή έκφραση πίνακα, "MaxAppearances," προσδιορίζει τον μέγιστο αριθμό εμφανίσεων μεταξύ όλων των μαγείρων. Τέλος, το κύριο ερώτημα επιλέγει το Αναγνωριστικό του Μάγειρα, το Όνομα του Μάγειρα, το Επώνυμό του και τον αριθμό των εμφανίσεών του από τον πίνακα των Μαγείρων, συνδέοντάς τον με τις κοινές εκφράσεις πίνακα "CookAppearances" και "MaxAppearances" για να απομακρύνει τους μάγειρες των οποίων ο αριθμός εμφανίσεων ανήκει στους πέντε κορυφαίους σε σχέση με τον μέγιστο αριθμό εμφανίσεων μείον πέντε.

4.8 : 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

```
SELECT episode_name, year, COUNT(*) AS EquipmentCount
```

```
FROM (
   SELECT DISTINCT E.episode_name, E.year, RE.Equipment_ID

   FROM Episode AS E



   INNER JOIN Recipe_Equipment AS RE ON E.Recipe_ID = RE.Recipe_ID

) AS EpisodeEquipment

GROUP BY episode_name, year

ORDER BY EquipmentCount DESC

LIMIT 1;
```

Στο όγδοο ερώτημα απλώς εκμεταλλευτήκαμε το table των επεισοδίων και κάνοντας το εσωτερικό σύνδεσμο με το Recipe Equipment και μέσω αυτού επιστρέφουμε το επεισόδιο την χρονιά και το μέγιστο πλήθος των εξαρτημάτων, αφού έχουμε ομαδοποιήσει κατά επεισόδιο και χρονιά διαγωνισμού σε φθίνουσα σειρά EquipmentCount τυπώνουμε μόνο το πρώτο(LIMIT 1) δηλαδή το ζητούμενο.

4.9 : Λίστα με μέσο όρο αριθμού γραμμαρίων υδατανθράκων στο διαγωνισμό ανά έτος;

```
SELECT `Year`, AVG(Carbs) AS AverageCarbsPerYear

FROM NutritionalInfo

INNER JOIN Recipe ON NutritionalInfo.Recipe_ID = Recipe.Recipe_ID

INNER JOIN Episode ON Recipe.Recipe_ID = Episode.Recipe_ID

GROUP BY `Year`;
```

Εν συνεχεία σε αυτό το ερώτημα έχουμε υπολογίσει για κάθε χρόνο το μέσο ποσό υδατανθράκων των συνταγών που εκτελέστηκαν στην διάρκια του διαγωνισμού. Επομένως, αντιλαμβανόμαστε ότι χρειαζόμαστε ένα join μεταξύ των διατροφικών πληροφοριών με τις συνταγές και τα επεισόδια

και κάνουμε group με βάση το έτος.

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

```
WITH CuisineEntries AS (
        SELECT
                c.Cuisine_ID,
                e.Year,
                COUNT(*) AS Entry Count
        FROM
                Cuisine c
        JOIN
                Recipe r ON c.Cuisine_ID = r.Cuisine_ID
        JOIN
                Episode e ON r.Recipe_ID = e.Recipe_ID
        GROUP BY
                c.Cuisine_ID,
                e.Year
        HAVING
                Entry_Count >= 3
);
ConsecutiveYears AS (
        SELECT
                CE1.Cuisine_ID,
                                        CE1.Year AS Year1,
                CE1.Entry_Count AS Entry_Count_Year1,
                CE2.Year AS Year2,
                CE2.Entry_Count AS Entry_Count_Year2
        FROM
                CuisineEntries CE1
        JOIN
                CuisineEntries CE2 ON CE1.Cuisine_ID = CE2.Cuisine_ID
                                        AND CE1.Year = CE2.Year - 1
)
SELECT
        c.Cuisine_ID,
        c.Cuisine_Name
FROM
        Cuisine c
JOIN
        ConsecutiveYears cy ON c.Cuisine_ID = cy.Cuisine_ID
WHERE
        cy.Entry_Count_Year1 = cy.Entry_Count_Year2;
```

4.11 :Βρείτε τους top-5 κριτές που έχουν δώσει συνολικά την υψηλότερη βαθμολόγηση σε

ένα μάγειρα. (όνομα κριτή, όνομα μάγειρα και συνολικό σκορ βαθμολόγησης)

```
SELECT
        s.Judge_ID,
        s.Cook_ID,
        c.Cook_Name,
        c.Surname,
        SUM (s.Score_Value) AS Total_Score
FROM
        Score s
JOIN
        Cook c ON s.Cook_ID = c.Cook_ID
WHERE
        s.Judge_ID IN (SELECT DISTINCT Cook_ID FROM Episode WHERE Recipe_ID IS
NULL)
GROUP BY
        s.Judge_ID,
        s.Cook_ID,
        c.Cook_Name,
        c.Surname
ORDER BY
        Total_Score DESC
        LIMIT 5;
```

Αυτό το ερώτημα SQL υπολογίζει το συνολικό σκορ για κάθε κριτή σε όλους τους μάγειρες που έχουν συμμετάσχει σε επεισόδιακαι επιλέγονται οι πέντε κορυφαίοι κριτές βάσει αυτού του συνολικού σκορ. Στο ερώτημα, συνδέεται ο πίνακας Score με τον πίνακα Cook βάσει του Cook_ID, ενώ ο περιορισμός των κριτών γίνεται μέσω του υποερωτήματος που επιλέγει τους μάγειρες που έχουν συμμετάσχει σε επεισόδια όπου δεν υπάρχει συνταγή. Τα αποτελέσματα ομαδοποιούνται ανά Judge_ID και Cook_ID και ταξινομούνται κατά φθίνουσα σειρά βάσει του συνολικού σκορ, ενώ επιλέγονται τα πρώτα πέντε αποτελέσματα

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

```
WITH ranked_recipes AS (

        SELECT

        SUM(r.difficulty_level) AS total_difficulty, e.Year, e.Episode_Name

        FROM Episode AS e

        INNER JOIN Recipe AS r ON e.Recipe_ID = r.Recipe_ID

        where e.Recipe_ID IS NOT NULL

        GROUP BY e.Year, e.Episode_Name

        ),
```

```
max_ranked_recipe AS (

        SELECT

        MAX(total_difficulty) AS max_total_difficulty

        FROM ranked_recipes

        )

SELECT *

FROM ranked_recipes AS rr

INNER JOIN max_ranked_recipe AS mrr

ON rr.total_difficulty = mrr.max_total_difficulty;
```

Αυτό το ερώτημα SQL χρησιμοποιεί δύο κοινές εκφράσεις πινάκων (CTEs) για να υπολογίσει το συνολικό επίπεδο δυσκολίας για κάθε επεισόδιο και να εντοπίσει το επεισόδιο με το μέγιστο επίπεδο δυσκολίας. Η πρώτη CTE, "ranked_recipes," υπολογίζει το συνολικό επίπεδο δυσκολίας ανά επεισόδιο συνδυάζοντας τον πίνακα Episode με τον πίνακα Recipe βάσει του Recipe_ID. Η δεύτερη CTE, "max_ranked_recipe," επιλέγει το μέγιστο συνολικό επίπεδο δυσκολίας ανάμεσα στα επεισόδια. Το κύριο ερώτημα συνδυάζει τις πληροφορίες αυτές για να επιστρέψει το επεισόδιο με το μέγιστο επίπεδο δυσκολίας, συνδυάζοντας τις δύο CTEs βάσει του συνολικού επιπέδου δυσκολίας.

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

```
WITH ranked_episodes AS (

        SELECT
        e.Year,
        e.Episode_Name,

        SUM(CASE

                WHEN c.Rank = "C' Cook" THEN 1

                WHEN c.Rank = "B' Cook" THEN 2

                WHEN c.Rank = "A' Cook" THEN 3

                WHEN c.Rank = 'Sous-Chef' THEN 4

                WHEN c.Rank = 'Head Chef' THEN 5

                ELSE 0
```

```
        END) AS total_rank
        FROM Episode AS e
        INNER JOIN Cook AS c ON e.Cook_ID = c.Cook_ID
        GROUP BY e.Year, e.Episode_Name
        ),
min_ranked_episodes AS (
        SELECT
        MIN(total_rank) AS min_total_rank
        FROM ranked_episodes
        )
SELECT re.*
FROM ranked_episodes AS re
INNER JOIN min_ranked_episodes AS mre
ON re.total_rank = mre.min_total_rank;
```

Αυτό το SQL ερώτημα χρησιμοποιεί δύο κοινές εκφράσεις πίνακα (CTEs) για να υπολογίσει το συνολικό "επίπεδο" κάθε επεισοδίου, λαμβάνοντας υπόψη τη βαθμολογία του μάγειρα σε κάθε επεισόδιο. Η πρώτη CTE, "ranked_episodes," υπολογίζει το συνολικό "επίπεδο" για κάθε επεισόδιο συνδυάζοντας τον πίνακα Episode με τον πίνακα Cook βάσει του Cook_ID. Η δεύτερη CTE, "min_ranked_episodes," επιλέγει το επεισόδιο με το ελάχιστο συνολικό "επίπεδο". Το κύριο ερώτημα επιστρέφει το επεισόδιο με το ελάχιστο "επίπεδο", συνδυάζοντας τις δύο CTEs βάσει του συνολικού "επιπέδου". Για την πραγμάτωση των παραπάνω χρησιμοποιήσαμε ένα mapping των ranks των μαγείρων σε ακεραίους αναθέτοντας 5 στον chef κι 1 στον c_cook.

4.14 :Ποια θεματική ενότητα έχει εμφανιστεί τις περισσότερες φορές στο διαγωνισμό;

```
SELECT
        t.Theme_Name,
        COUNT (rt.Theme_ID) AS Appearance_Count
FROM
        Rec_Them rt
JOIN
        Theme t ON rt.Theme_ID = t.Theme_ID
GROUP BY
        rt.Theme_ID, t.Theme_Name
```

```
ORDER BY
    Appearance_Count DESC
LIMIT 1;
```

Αυτό το ερώτημα SQL υπολογίζει το θέμα που έχει εμφανιστεί τις περισσότερες φορές στις συνταγές.Συγκεκριμένα, συνδέει τον πίνακα των σχέσεων συνταγών με θέματα (Rec_Them) με τον πίνακα των θεμάτων (Theme) βάσει του αναγνωριστικού του θέματος. Έπειτα, ομαδοποιεί τα αποτελέσματα ανά αναγνωριστικό θέματος και ονομασία θέματος, υπολογίζοντας τον αριθμό των εμφανίσεων κάθε θέματος. Τέλος, ταξινομεί τα αποτελέσματα κατά φθίνουσα σειρά βάσει του αριθμού εμφανίσεων και επιστρέφει το πρώτο αποτέλεσμα, που αντιστοιχεί στο θέμα με τις περισσότερες εμφανίσεις.

4.15 :Ποιες ομάδες τροφίμων δεν έχουν εμφανιστεί ποτέ στον διαγωνισμό;

```
SELECT f.FoodGroup_Name
FROM FoodGroup AS f
WHERE f.FoodGroup_ID NOT IN (
    SELECT ci.FoodGroup_ID
    FROM episode AS e
    INNER JOIN ingr_rec AS ri ON e.Recipe_ID = ri.Recipe_ID
    INNER JOIN Ingredients AS ri ON ci.ingredients_ID = ri.ingredients_id
    GROUP BY ci.FoodGroup_id
);
```

Αυτό το ερώτημα SQL επιλέγει τις ονομασίες των ομάδων τροφίμων που δεν εμφανίζονται στα επεισόδια. Αρχικά, επιλέγει όλες τις ονομασίες ομάδων τροφίμων από τον πίνακα FoodGroup. Στη συνέχεια, χρησιμοποιεί ένα υποερώτημα για να επιλέξει τις FoodGroup_ID που δεν αντιστοιχούν σε συνταγές. Αυτό γίνεται με τη σύνδεση του πίνακα επεισοδίων με τον πίνακα ingr_rec και τον πίνακα συστατικών Ingredients, συγκεντρώνοντας τα στοιχεία βάσει του FoodGroup_ID. Τέλος, επιλέγονται οι ονομασίες ομάδων τροφίμων που δεν ανήκουν στο σύνολο των FoodGroup_ID που επιλέγονται από το υποερώτημα.

-Indexes-

Όπως είπαμε και παραπάνω, indexes δεν χρησιμοποιήθηκαν στα queries, κάτι το οποίο επιτράπηκε και από το μέγεθος της βάσης, αφού έτσι κι αλλιώς όλα γινόντουσαν αρκετά γρήγορα. Ωστόσο, μπορούμε να δώσουμε τώρα κάποια παραδείγματα χρήσης και αξιοποίησης των trigger:

```
CREATE INDEX idx_recipe_name ON Recipe (Name);
```

```
SELECT *
FROM Recipe
WHERE Name LIKE '%pasta%';
```

Απαίτηση 5: Οδηγίες χρήσης της εφαρμογής

5.1 Εγκατάσταση MySQL

Εγκατάσταση του XAMPP και του MySQL Workbench.

5.2 Λήψη απαραίτητων αρχείων από το GitHub

Στο σύνδεσμο: https://github.com/shmmytzi/Cooking_Competition, θα βρείτε τα απαραίτητα αρχεία (SQL

Scripts, Διαγράμματα κλπ) για τη δημιουργία της βάσης και τη γενικότερη χρήση της εφαρμογής.

5.3 Δημιουργία της βάσης

Ανοίγοντας και τρέχοντας στο MySQL Workbench τα αρχεία DDL και DML, η βάση είναι σχεδόν έτοιμη. Σε αυτό το σημείο, πρέπει να τρέξουμε και το CountCalories procedure.

5.4 Κλήρωση Διαγωνισμού

Η επιλογή των εθνικών κουζινών, των συνταγών, των διαγωνιζόμενων των κριτών, καθώς και η λήψη των σκορ, γίνονται με τη χρήση των αντίστοιχων procedures, τα οποία θα βρείτε και στο git repository. Οι συμμετέχοντες κληρώνονται ανά 10 επεισόδια, δηλαδή ανά έτος, ενώ τα σκορ ανά επεισόδιο.

5.5 Users

Το συγκεκριμένο κομμάτι της εφαρμογής μπορεί να υλοποιηθεί μέσω της χρήσης procedure και την αξιοποίηση των views. Λόγω κάποιον τεχνικών προβλημάτων η υλοποίηση δεν μπόρεσε να ολοκληρωθεί/επαληθευθεί. Ωστόσο, η κεντρική ιδέα είναι:

Procedure:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `Users`()
BEGIN
CREATE USER 'admin'@'localhost' IDENTIFIED BY 'secure_password';
GRANT ALL PRIVILEGES ON *.* TO 'admin'@'localhost' WITH GRANT OPTION;
FLUSH PRIVILEGES;

-- cook user
CREATE USER 'Cook'@'localhost' IDENTIFIED BY 'secure_password';
grant insert on Recipe to 'Cook'@'localhost';
grant all privileges on cook_info to 'Cook'@'localhost';
grant all privileges on cook_recipes to 'Cook'@'localhost';

END
```

Views:

```
CREATE VIEW `cook_info` AS
SELECT *
FROM Cook
WHERE Cook_ID='3121001';

CREATE VIEW `cook_recipes` AS
SELECT *
FROM Recipe
WHERE Recipe_ID IN (
   SELECT Recipe_ID
   FROM Episode
   WHERE Cook_ID = '3121001' AND Recipe_ID IS NOT NULL
);


GRANT SELECT, UPDATE ON `cook_info` TO 'Cook'@'localhost';
GRANT SELECT, INSERT, UPDATE ON `cook_recipes` TO 'Cook'@'localhost';
FLUSH PRIVILEGES;
```