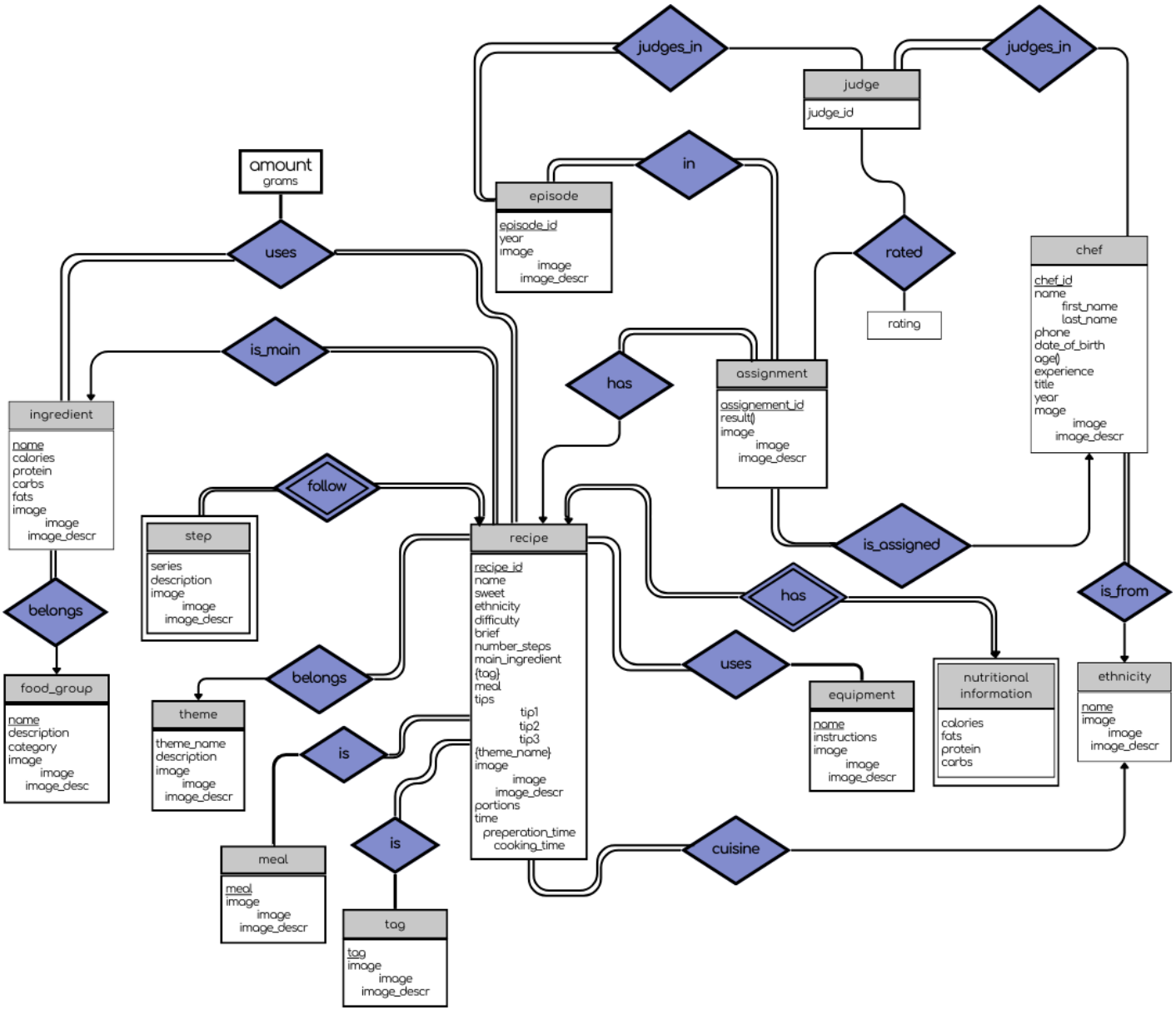
ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ
ΑΝΑΦΟΡΑ ΕΞΑΜΗΝΙΑΙΑΣ ΕΡΓΑΣΙΑΣ

Ομάδα 115: Χαράλαμπος Παπαδόπουλος 03120199
Κυριακή Φραγκονικολάκη 03121215
Ειρήνη Στρουμπάκου 03121183

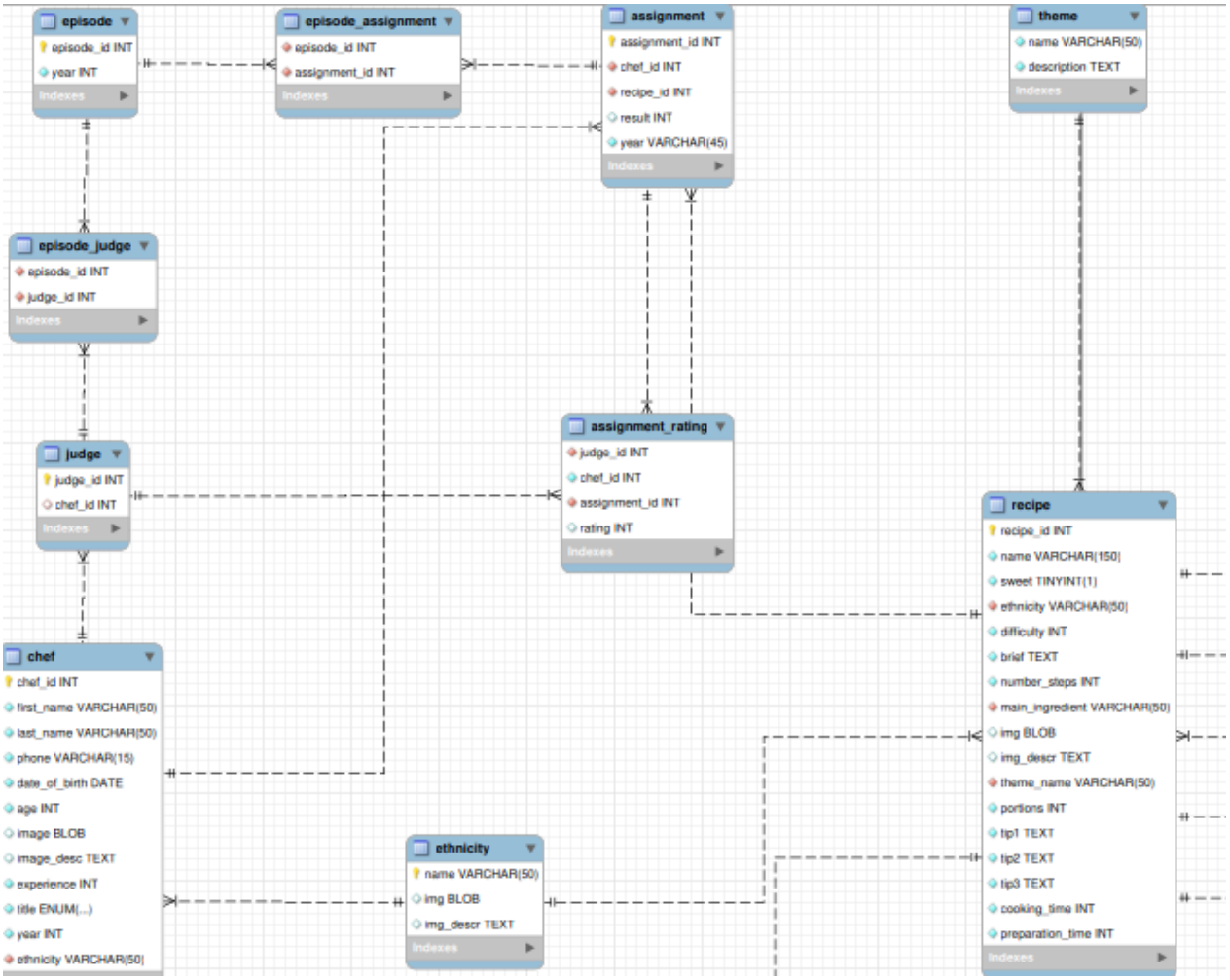
https://github.com/harrispapad/DB-project/tree/main

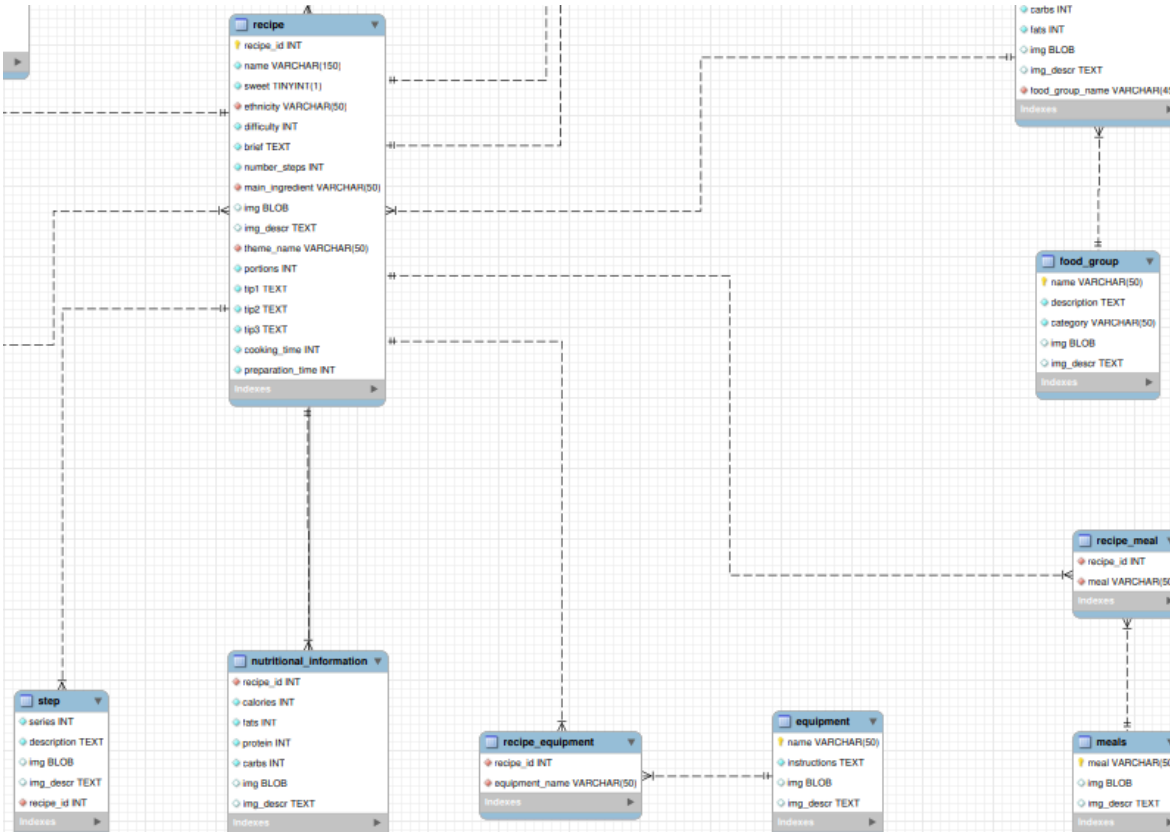
1) Διάγραμμα ER:

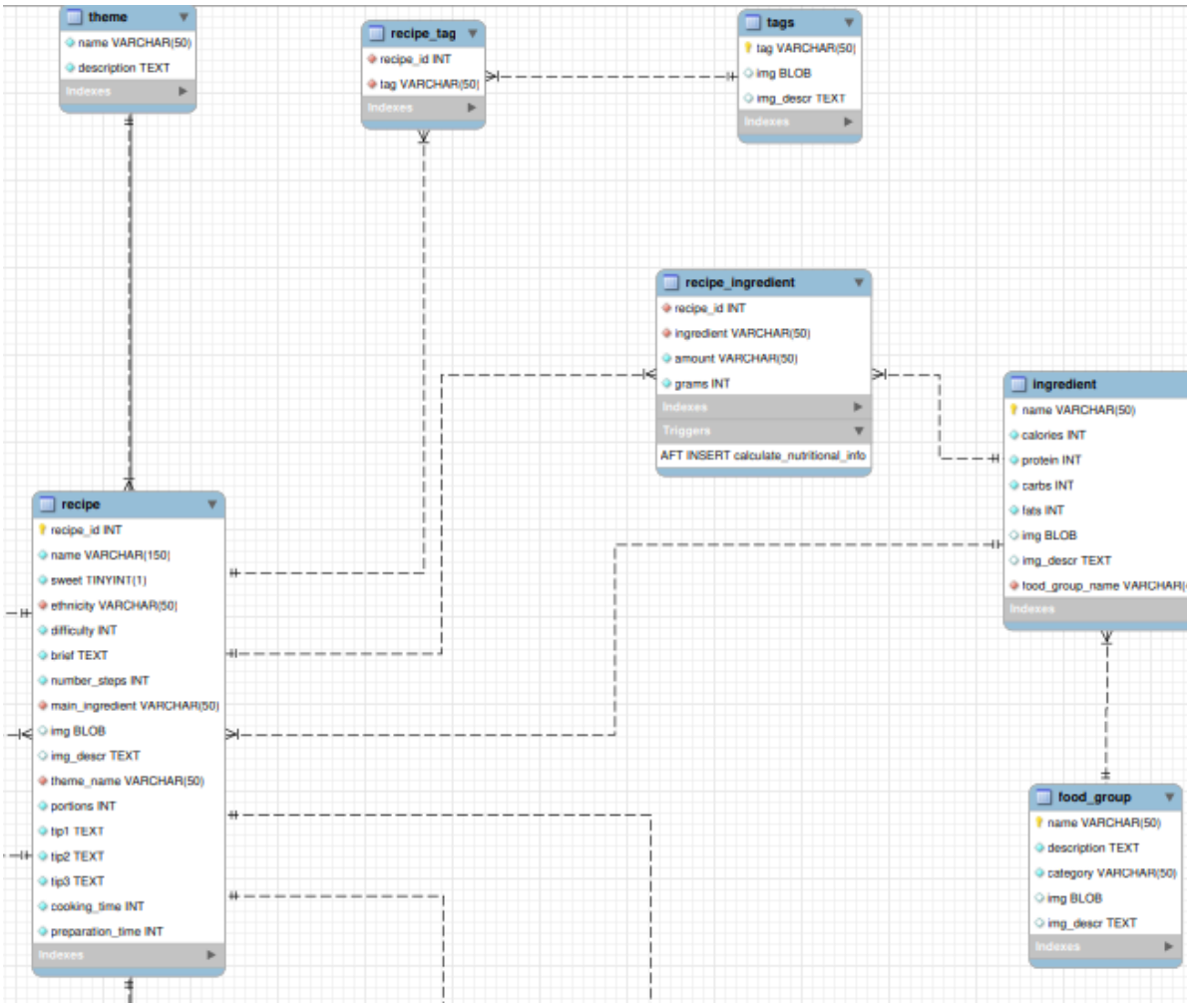


2) Σχεσιακό Διάγραμμα:

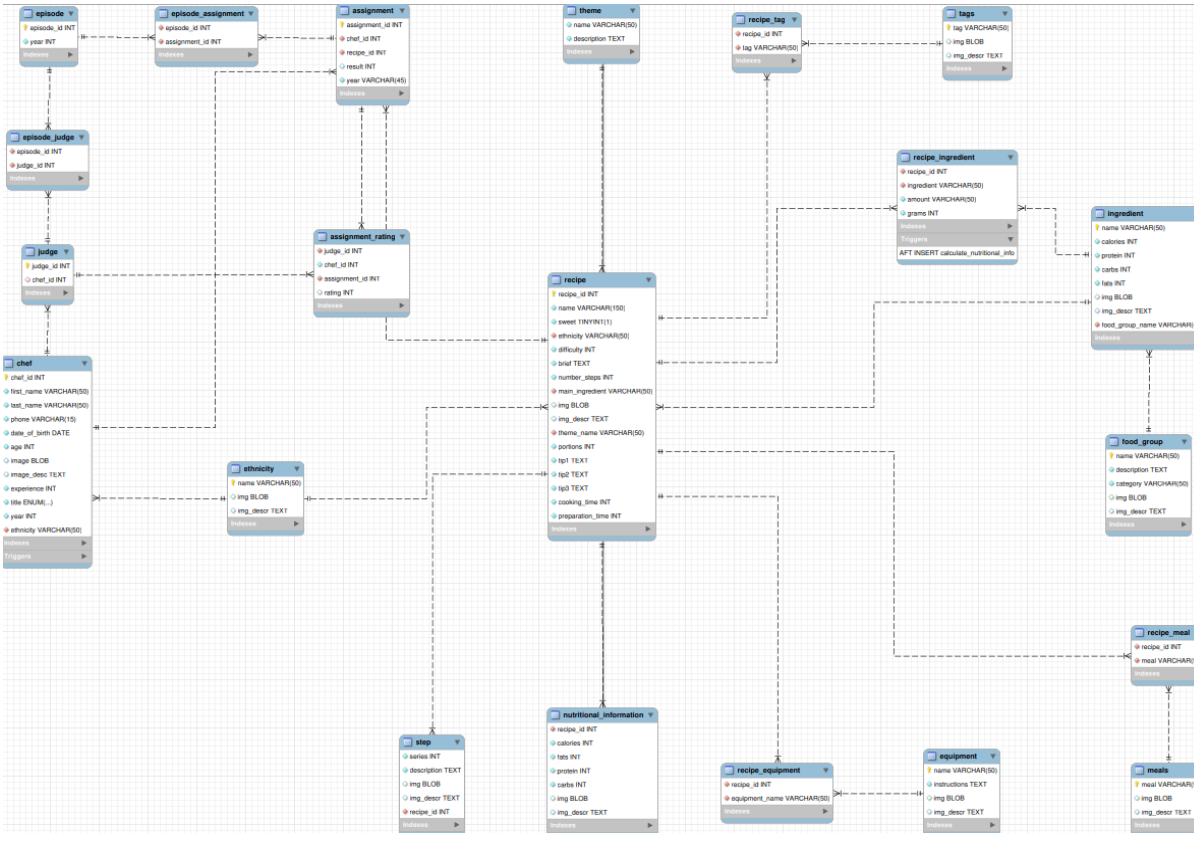
Το παραθέτουμε σε διάφορα τμήματα, όλα περικομμένα γύρω από τον πίνακα “recipe”, έτσι ώστε να είναι πιο ευκρινές.







Και συνολικά:



3) Σχολιασμός Σχεσιακού Διαγράμματος:

Στην βάση μας έχουμε τρία βασικά entities: τις συνταγές(recipe), τους μάγειρες(chef) και την ανάθεση συνταγών(assignments). Έπειτα, το recipe συνδέεται με το recipe ingredient, το οποίο με την σειρά του συνδέεται με το ingredient, προκειμένου έτσι να έχουμε ξεχωριστά τις απαραίτητες πληροφορίες για κάθε υλικό της συνταγής μας, αφού μία συνταγή έχει άγνωστο αριθμό υλικών και κάθε υλικό μπορεί να χρησιμοποιηθεί σε πολλές συνταγες. Ομοίως λειτουργούμε για τους πίνακες equipment, meal, tag και recipe_equipment, recipe_meal, recipe_tag αντίστοιχα. Επιπλέον, το recipe συνδέεται με την οντότητα ethnicity, η οποία ωστόσο ενώνεται και με τους chef, καθώς αφενός κάθε συνταγή ανήκει σε μία συγκεκριμένη εθνική κουζίνα και αφετέρου κάθε μάγειρας ειδικεύεται σε μία εθνική κουζίνα. Τέλος, εφόσον κάθε συνταγή ανήκει σε συγκεκριμένη θεματική κουζίνα και όχι σε πολλές ταυτόχρονα, υλοποιήσαμε το entity theme μόνο και δεν χρειαστήκαμε όπως στις άλλες κατηγορίες recipe_theme. Όσον αφορά το άλλο βασικό entity, δηλαδή τους chef, συνδέονται με μια οντότητα judge, καθώς έχουν την δυνατότητα πέρα από μάγειρες σε κάποια επεισόδια να είναι και κριτές. Για να καθορίσουμε ωστόσο πότε είναι κριτές, ορίζουμε νέα οντότητα το episode_judge, που συνδέει τα επεισόδια με τους κριτές. Τέλος, η ανάθεση των επεισοδίων και των assignments(δηλαδή ποια συνταγή αναλαμβάνει κάθε συμμετέχοντας σε κάθε επεισόδιο), υπάρχουν τα episode, episode_assignment, assignment και assignment_rating για την βαθμολόγηση των διαγωνιζόμενων.

4) Indexes:

Στην βάση μας κάνουμε χρήση ευρετηρίων για την ταχύτερη ανάκτηση δεδομένων.
Αυτά έχουν σχεδιαστεί με βάση τα attributes τα οποία χρησιμοποιούνται συχνότερα στα queries, joins και wheres. Αυτά είναι κυρίως τα primary keys όπως το chef_id και το recipe_id, στα οποία λόγω του παραπάνω constraint τους, ορίζεται αυτόματα το index τους.
Επιπλέον, κάνουμε συχνή χρήση των πεδίων με foreign keys τα οποία όταν είναι παράλληλα σε άλλον πίνακα ως primary keys, έχουν ήδη ευρετήριο ενώ αν όχι, το τοποθετούμε εμείς. Αυτό συνέβη για το “name” στον πίνακα “theme” το οποίο είναι foreign key στον πίνακα “recipe”:

```
CREATE INDEX idx_name ON theme(name);
```

Συγκεκριμένα για τα queries η μόνη προσθήκη ευρετηρίου που δεν καλύπτεται από τα παραπάνω, ήταν για τα “first_name” και “last_name” στον πίνακα “chef” τα οποία χρησιμοποιούνται στα 3.1, 3.2 ερωτήματα.

```
CREATE INDEX idx_first_name ON chef(first_name);
CREATE INDEX idx_last_name ON chef(last_name);
```

5) Constraints:

1. Κάθε συνταγή(recipe) μπορεί να ανήκει μόνο σε μία θεματική ενότητα(theme)
2. Κάθε κριτής(judge) πρέπει να έχει και chef_id, αφού σε άλλα επεισόδια μπορεί να συμμετέχει ως μάγειρας. Αυτο το εξασφαλίζουμε με το entity judge, το οποίο συνδέει τους κριτές με τους chef και περιλαμβάνει chef_id και judge_id ξεκαθαρίζοντας ότι πρόκειται για δύο ξεχωριστές έννοιες.
3. Κάθε υλικό(ingredient) ανήκει μόνο σε μία ομάδα τροφίμων(food_group) από τις 12 προκαθορισμένες.
4. Σαν ειδίκευση(title) στους μάγειρες(chef) έχουμε 5 συγκεκριμένες τιμές: A cook, B cook, C cook, chef, sous chef.
5. Η ειδίκευση σε εθνικότητα ενός σεφ, δεν μπορεί να είναι διαφορετική από τις πιθανές εθνικότητες των συνταγών.
6. Η βαθμολογία των κριτών παίρνει τιμές από 1 έως 5.
7. Η ηλικία κάθε μάγειρα(age) πρέπει να είναι μεγαλύτερη του 0.
8. Η δυσκολία των συνταγών είναι σε κλίμακα 1 έως 5 με 1 το πιο εύκολο και 5 το πιο δύσκολο.

6) Procedures:

Για κάποιες από τις παρακάτω διαδικασίες χρειαστήκαμε τις mysql.connector, random.

1. Επαγγελματική κατάρτιση σε συγκρίσιμη τιμή:
   Μετατρέψαμε τον τίτλο επαγγελματικής κατάρτισης των chef (πχ sous chef, A cook) σε αριθμούς από το 1 έως το 5, με 5 να είναι ο περισσότερο έμπειρος και 1 ο πιο αρχάριος. Έτσι ήταν συγκρίσιμες τιμές το οποίο βοήθησε στο query 3.13.

```
ALTER TABLE chef
ADD COLUMN expertise_level INT;
ALTER TABLE chef
ADD CONSTRAINT CHECK (expertise_level BETWEEN 1 AND 5);


CREATE PROCEDURE UpdateChefExpertiseLevel()
BEGIN
   UPDATE chef
   SET expertise_level = CASE
      WHEN title = 'chef' THEN 5
      WHEN title = 'sous chef' THEN 4
      WHEN title = 'A cook' THEN 3
      WHEN title = 'B cook' THEN 2
      WHEN title = 'C cook' THEN 1
      ELSE NULL
   END;
END //

CALL UpdateChefExpertiseLevel();
```

2. Υπολογισμός τελικής βαθμολογίας πιάτου:
   Κάθε κριτής που έχει ανατεθεί σε ένα επεισόδιο (συνολικά είναι τρεις), έχει μία βαθμολογία για κάθε συνταγή που εμφανίστηκε στο επεισόδιο. Με την παρακάτω διαδικασία υπολογίζεται η τελική βαθμολογία του πιάτου που παρουσιάζει ένας chef η

οποία είναι ο μέσος όρος των τριων βαθμολογιών που έλαβε. Έπειτα, αυτή η βαθμολογία αποθηκεύεται ως το “result” στον πίνακα “assignment”.

```
PROCEDURE `calculate_assignment_rating`()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE assignment_id_val INT;
    DECLARE rating_sum DECIMAL(5, 2);
    DECLARE rating_avg DECIMAL(5, 2);

    -- Declare cursor to fetch assignment IDs where rating is null
    DECLARE assignment_cursor CURSOR FOR
        SELECT assignment_id FROM assignment WHERE result IS NULL;

    -- Declare continue handler for cursor
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN assignment_cursor;

    assignment_loop:LOOP
        FETCH assignment_cursor INTO assignment_id_val;
        IF done THEN
            LEAVE assignment_loop;
        END IF;

        -- Calculate average rating for the assignment
        SELECT (SUM(rating) / 3) INTO rating_sum
        FROM assignment_rating
        WHERE assignment_id = assignment_id_val;

        SET rating_avg = IFNULL(rating_sum, 0);

        -- Update assignment table with the calculated average rating
        UPDATE assignment
        SET result = rating_avg
        WHERE assignment_id = assignment_id_val;
    END LOOP;

    CLOSE assignment_cursor;
END ;;
```

3. Κλήρωση και δημιουργία των αναθέσεων σεφ-συνταγής:
Στην παρακάτω διαδικασία, δημιουργούνται τα assignments δηλαδή τα “ζεύγη” σεφ-συνταγών τα οποία θα εμφανιστούν στα επεισόδια. Θεωρούμε ότι ένας σεφ θα μπορεί να μαγειρέψει συνταγές που ανήκουν στην κουζίνα της εθνικότητάς του.

```
cursor = conn.cursor()

def generate_assignments(cursor, conn, num_years):
    for year in range(1, num_years + 1):
        for assignment_id in range(year * 100, (year + 1) * 100):
            # Select an ethnicity for the assignment
```

```
            cursor.execute("SELECT name FROM ethnicity ORDER BY RAND()
LIMIT 1")
            ethnicity_row = cursor.fetchone()
            if ethnicity_row:
                ethnicity = ethnicity_row[0]  # Extract the value from the tuple
            else:
                print(f"No ethnicity found at assignment {assignment_id} for year {year}")
                continue  # Skip this iteration if no ethnicity is found

            # Select chef
             cursor.execute("SELECT chef_id FROM chef WHERE ethnicity = %s and
year = %s ORDER BY RAND() LIMIT 1", (ethnicity, year))
            chef_row = cursor.fetchone()
            if chef_row:
                chef_id = chef_row[0]  # Extract the value from the tuple
            else:
                                print(f"No chef found for ethnicity {ethnicity} at assignment
{assignment_id} for year {year}")
                continue  # Skip this iteration if no chef is found

           # Select one recipe per ethnicity
             cursor.execute("SELECT recipe_id FROM recipe WHERE ethnicity = %s
ORDER BY RAND() LIMIT 1", (ethnicity,))
            recipe_row = cursor.fetchone()
            if recipe_row:
                recipe_id = recipe_row[0]  # Extract the value from the tuple
            else:
                               print(f"No recipe found for ethnicity {ethnicity} at assignment
{assignment_id} for year {year}")
                continue  # Skip this iteration if no recipe is found

            # Insert the assignment into the "assignment" table
             sql = "INSERT INTO assignment (assignment_id, chef_id, recipe_id, year)
VALUES (%s, %s, %s, %s)"
            cursor.execute(sql, (assignment_id, chef_id, recipe_id, year))
            conn.commit()

generate_assignments(cursor, conn, 2)  # Call the function for 3 years

cursor.close()
conn.close()
```

4. Φόρτωση των assignments σε επεισόδια:
   Με τον παρακάτω τρόπο αντιστοιχίσαμε τα “ζεύγη” chef-recipe σε επεισόδια. Κάθε επεισόδιο περιέχει δέκα τέτοια ζεύγη και κάθε χρονιά κατά την οποία συμβαίνει ο διαγωνισμός μαγειρικής έχει δέκα επεισόδια. Τέλος, υποθέτουμε ότι ο διαγωνισμός “τρέχει” για τρεις χρονιές.

```
def load_ethnicities(cursor):
    cursor.execute("SELECT name FROM ethnicity")
    return [row[0] for row in cursor.fetchall()]
```

```
# Function to load all assignments for a given year
def load_assignments(cursor, year):
     cursor.execute("SELECT assignment_id FROM assignment WHERE year = %s", (year,))
   return [row[0] for row in cursor.fetchall()]

# Function to create an episode with 10 assignments, each with a unique ethnicity
def create_episode_assignment(cursor, conn, episode_id, year, all_assignments, ethnicities):
   if len(all_assignments) < 10:
      print("Not enough assignments available for the year")
      return

   if len(ethnicities) < 10:
      print("Not enough unique ethnicities available")
      return

   # Randomly select 10 assignments from the loaded assignments
   selected_assignments = random.sample(all_assignments, 10)

   # Randomly select 10 unique ethnicities
   selected_ethnicities = random.sample(ethnicities, 10)

   # Insert the episode assignments into the database
   for assignment_id in selected_assignments:
         sql = "INSERT INTO episode_assignment (episode_id, assignment_id) VALUES (%s, %s)"
      cursor.execute(sql, (episode_id, assignment_id))
         all_assignments.remove(assignment_id)  # Remove the assignment from the list to prevent duplication


   conn.commit()

# Main function
def main():
   conn = create_connection()
   cursor = conn.cursor()

   # Load all ethnicities
   ethnicities = load_ethnicities(cursor)

   # Define the years for the episodes and assignments
   for year in range(1, 3):
      # Load all assignments for the given year
      all_assignments = load_assignments(cursor, year)

      # Create 10 episodes for the current year
      for episode_id in range(1, 11):
         # Adjust episode_id based on the year
         adjusted_episode_id = year * 100 + episode_id

         # Create episode assignments
```

```
            create_episode_assignment(cursor, conn, adjusted_episode_id, year,
all_assignments, ethnicities)

    cursor.close()
    conn.close()
```

5. Ανάθεση τριών κριτών σε κάθε επεισόδιο:

```
# Function to load all assignments for a given episode
def load_assignments(cursor, episode_id):
     cursor.execute("SELECT assignment_id FROM episode_assignment WHERE
episode_id = %s", (episode_id,))
    return [row[0] for row in cursor.fetchall()]

def create_episode_judge(cursor, episode_id, year):
    # Load episode assignments
    episode_assignments = load_assignments(cursor, episode_id)

    # Find the chefs of the current episode
    episode_chefs = []
    for assignment_id in episode_assignments:
         cursor.execute("SELECT chef_id from assignment where assignment_id =
%s", (assignment_id,))
        chef_id = cursor.fetchone()[0]  # Fetch the chef_id
        episode_chefs.append(chef_id)  # Add chef_id to the list

    # Fetch years chefs
    cursor.execute("SELECT chef_id FROM chef WHERE year = %s", (year,))
    year_chefs = [row[0] for row in cursor.fetchall()]

    # Filter out chefs that are already cooking in this episode
        available_chefs = [chef_id for chef_id in year_chefs if chef_id not in
episode_chefs]

    # Choose three random chefs from the available chefs list
    random_judges = random.sample(available_chefs, min(3, len(available_chefs)))

    # Insert judges into the episode_judge table
    for judge in random_judges:
            cursor.execute("INSERT INTO episode_judge (episode_id, judge_id)
VALUES (%s, %s)", (episode_id, (judge * 10)))
  def main():
    conn = create_connection()
    cursor = conn.cursor()

    for year in range(1, 3):
       for episode in range(year * 100 + 1, year * 100 + 11):
         create_episode_judge(cursor, episode, year)

    conn.commit()
    cursor.close()
    conn.close()
main()
```

7) DDL & DML scripts

Παρακάτω παραθέτουμε ως παράδειγμα την δημιουργία των tables recipe και chef:

```
CREATE TABLE `recipe` (
  `recipe_id` int NOT NULL,
   `name` varchar(150) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
  `sweet` tinyint(1) NOT NULL,
   `ethnicity` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
  `difficulty` int NOT NULL,
  `brief` text CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
  `number_steps` int NOT NULL,
        `main_ingredient` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
  `img` blob,
  `img_descr` text CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci,
   `theme_name` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
  `portions` int NOT NULL,
  `tip1` text CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
  `tip2` text CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci,
  `tip3` text CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci,
  `cooking_time` int NOT NULL,
  `preparation_time` int NOT NULL,
  PRIMARY KEY (`recipe_id`),
  KEY `idx_recipe_main_ingredient` (`main_ingredient`),
  KEY `idx_recipe_theme_name` (`theme_name`),
  KEY `fk_ehtnicity_idx` (`ethnicity`),
    CONSTRAINT `fk_recipe_ethnicity` FOREIGN KEY (`ethnicity`) REFERENCES `ethnicity` (`name`) ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT `fk_recipe_main_ingredient` FOREIGN KEY (`main_ingredient`) REFERENCES `ingredient` (`name`) ON DELETE RESTRICT ON UPDATE CASCADE,
   CONSTRAINT `fk_recipe_theme` FOREIGN KEY (`theme_name`) REFERENCES `theme` (`name`) ON DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT `recipe_chk_1` CHECK (((`difficulty` >= 1) and (`difficulty` <= 5))),
  CONSTRAINT `recipe_chk_2` CHECK ((`portions` >= 0)),
  CONSTRAINT `recipe_chk_3` CHECK ((`portions` >= 0))
)
```

```
CREATE TABLE `chef` (
  `chef_id` int NOT NULL AUTO_INCREMENT,
   `first_name` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
   `last_name` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
   `phone` varchar(15) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
  `date_of_birth` date NOT NULL,
```

```
  `age` int NOT NULL,
  `image` blob,
  `image_desc` text CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci,
  `experience` int NOT NULL,
    `title` enum('C cook','B cook','A cook','sous chef','chef') CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci NOT NULL,
  `year` int NOT NULL,
    `ethnicity` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL,
  PRIMARY KEY (`chef_id`),
  KEY `fk_chef_ethinicity` (`ethnicity`),
   CONSTRAINT `fk_chef_ethnicity` FOREIGN KEY (`ethnicity`) REFERENCES `ethnicity`
(`name`) ON DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT `chef_chk_1` CHECK ((`experience` >= 0))
)
```

Παραθέτουμε και δύο ενδεικτικά παραδείγματα προσθήκης εικόνας κάνοντας χρήση link:

```
UPDATE recipes
SET img_descr = 'spinach with eggs',
    img_url =
'https://img.jamieoliver.com/jamieoliver/recipe-database/156460151.jpg?tr=w-800,h-1066'
WHERE name = 'spinach with eggs';
UPDATE recipes
SET img_descr = 'creme caramele',
    img_url =
'https://img.jamieoliver.com/jamieoliver/recipe-database/155517325.jpg?tr=w-800,h-1066'
WHERE name = 'creme caramele';
```

8) Queries:

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

   α) Ανά μάγειρα:

```
SELECT chef_id, ROUND(AVG(result), 2) AS avg_result
FROM assignment
GROUP BY chef_id
ORDER BY avg_result DESC;
```

   β) Ανά εθνική κουζίνα:

```
SELECT sq.ethnicity, ROUND(AVG(sq.result), 2) as avg_result
from (
   SELECT c.ethnicity, a.result
        FROM chef c
        JOIN assignment a ON c.chef_id = a.chef_id) as sq
GROUP BY sq.ethnicity
ORDER BY avg_result DESC;
```

2. Για δεδομένη Εθνική κουζίνα και έτος, ποιοι μάγειρες ανήκουν σε αυτήν και ποιοι μάγειρες συμμετείχαν σε επεισόδια;
   Εδώ θεωρήσαμε ότι ουσιαστικά θέλουμε δύο ξεχωριστά αποτελέσματα: α) ποιοι μάγειρες ανήκουν στην δεδομένη εθνική κουζίνα και β) ποιοι μάγειρες με ειδίκευση στην δεδομένη εθνική κουζίνα συμμετείχαν σε επεισόδια του δεδομένου έτους

```
SELECT
    'Wanted ethinicity and participated' AS TableName,
    c.ethnicity AS c_ethnicity,
    c.first_name AS c_first_name,
    c.last_name AS c_last_name,
    c.year AS c_year
FROM chef c
WHERE c.ethnicity = 'portugal'
  AND c.chef_id IN (SELECT chef_id FROM assignment)

UNION ALL

SELECT
    'Wanted ethnicity and wanted year' AS TableName,
    b.ethnicity AS b_ethnicity,
    b.first_name AS b_first_name,
    b.last_name AS b_last_name,
    b.year AS b_year
FROM chef b
WHERE b.ethnicity = 'portugal'
  AND b.year = 1 ;
```

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

```
SELECT a.chef_id, count(*)
FROM assignment a
JOIN chef c ON a.chef_id = c.chef_id
WHERE c.age < 30
GROUP BY a.chef_id;
```

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

```
SELECT judge_id
FROM judge
WHERE judge_id not in (
        SELECT judge_id
        FROM episode_judge);
```

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

Τρέξαμε αρχικά το query με having count(*) > 3, αλλά παρατηρούμε ότι κανένας κριτής δεν έχει συμμετάσχει στον ίδιο χρόνο πάνω από 3 φορές. Επομένως, τρέχουμε το query μας με having count(*) > 2 για να εμφανιστούν αποτελέσματα.

```
WITH yearly_appearances AS (
   SELECT judge_id, COUNT(*) AS c
   FROM episode_judge
   GROUP BY judge_id
   HAVING COUNT(*) > 2
)
SELECT a.judge_id as 'judge a', b.judge_id as 'judge b', a.c as 'amount of appearances'
FROM yearly_appearances a
JOIN yearly_appearances b ON a.c = b.c AND a.judge_id <> b.judge_id
ORDER BY a.c DESC, a.judge_id;
```

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

```
SELECT tag_pair, COUNT(*) AS pair_count
FROM (
   SELECT
      CONCAT(rt1.tag, ',', rt2.tag) AS tag_pair
   FROM recipe_tag rt1
   JOIN recipe_tag rt2 ON rt1.recipe_id = rt2.recipe_id
   JOIN recipe r1 ON rt1.recipe_id = r1.recipe_id
   JOIN recipe r2 ON rt2.recipe_id = r2.recipe_id
   JOIN assignment a ON r1.recipe_id = a.recipe_id
   WHERE rt1.tag < rt2.tag
) AS tag_pairs
GROUP BY tag_pair
ORDER BY pair_count DESC
LIMIT 3;
```

και query plan

```
EXPLAIN
SELECT tag_pair, COUNT(*) AS pair_count
FROM (
   SELECT
      CONCAT(rt1.tag, ',', rt2.tag) AS tag_pair
   FROM recipe_tag rt1 FORCE INDEX (rt_recipe_id_index)
     JOIN recipe_tag rt2 FORCE INDEX (rt_recipe_id_index) ON rt1.recipe_id = rt2.recipe_id
       JOIN recipe r1 FORCE INDEX (r_recipe_id_index) ON rt1.recipe_id = r1.recipe_id
       JOIN recipe r2 FORCE INDEX (r_recipe_id_index) ON rt2.recipe_id = r2.recipe_id
     JOIN assignment a FORCE INDEX (a_recipe_id_index) ON r1.recipe_id = a.recipe_id
   WHERE rt1.tag < rt2.tag
) AS tag_pairs
GROUP BY tag_pair
```

```
ORDER BY pair_count DESC
LIMIT 3;
```

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

```
SELECT a.chef_id, count(*) as times
FROM assignment a
JOIN chef c ON a.chef_id = c.chef_id
GROUP BY a.chef_id
HAVING times < (
   SELECT MAX(times) - 5
      FROM (
         SELECT COUNT(*) AS times
         FROM assignment
         GROUP BY chef_id
      ) AS subquery
);
```

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

```
SELECT ep_as.episode_id, COUNT(re.equipment_name) AS total_equipment
FROM episode_assignment ep_as
JOIN assignment a ON ep_as.assignment_id = a.assignment_id
JOIN recipe_equipment re ON a.recipe_id = re.recipe_id
GROUP BY ep_as.episode_id
ORDER BY total_equipment
LIMIT 1;
```

και query plan

```
EXPLAIN SELECT ep_as.episode_id, COUNT(re.equipment_name) AS
total_equipment
FROM episode_assignment ep_as
JOIN assignment a FORCE INDEX (idx_assignment_id) ON
ep_as.assignment_id = a.assignment_id
JOIN recipe_equipment re ON a.recipe_id = re.recipe_id
GROUP BY ep_as.episode_id
ORDER BY total_equipment
LIMIT 1;
```

9. Λίστα με μέσο όρο αριθμού γραμμαρίων υδατανθράκων στο διαγωνισμό ανά έτος;

```
SELECT avg(c.carbs), b.year
FROM nutritional_information c
join assignment b ON c.recipe_id = b.recipe_id
WHERE b.year in (select distinct year from assignment)
group by b.year;
```

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

```
SELECT distinct r1.ethnicity, r2.ethnicity
```

```
FROM recipe r1
JOIN assignment a1 ON a1.recipe_id = r1.recipe_id
JOIN episode_assignment ea1 ON ea1.assignment_id = a1.assignment_id
JOIN episode e1 ON ea1.episode_id = e1.episode_id
JOIN recipe r2 ON r1.ethnicity < r2.ethnicity
JOIN assignment a2 ON a2.recipe_id = r2.recipe_id
JOIN episode_assignment ea2 ON ea2.assignment_id = a2.assignment_id
JOIN episode e2 ON ea2.episode_id = e2.episode_id
WHERE
   (SELECT COUNT(*)
    FROM assignment a1
    JOIN recipe r ON a1.recipe_id = r.recipe_id
    WHERE r.ethnicity = r1.ethnicity) =
   (SELECT COUNT(*)
    FROM assignment a2
    JOIN recipe r ON a2.recipe_id = r.recipe_id
    WHERE r.ethnicity = r2.ethnicity)
   AND ABS(e1.year - e2.year) = 1;
```

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 judge_id, chef_id, total_rating
FROM (
   SELECT judge_id, chef_id, SUM(rating) as total_rating,
          ROW_NUMBER() OVER (PARTITION BY chef_id ORDER BY SUM(rating)
DESC) AS rating_rank
   FROM assignment_rating
   GROUP BY judge_id, chef_id
) AS ranked_ratings
WHERE rating_rank <= 5
ORDER BY chef_id, total_rating DESC
```

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

```
SELECT ep_as.episode_id, SUM(r.difficulty) AS total_difficulty
FROM episode_assignment ep_as
JOIN assignment a ON ep_as.assignment_id = a.assignment_id
JOIN recipe r ON a.recipe_id = r.recipe_id
GROUP BY ep_as.episode_id
ORDER BY total_difficulty DESC
LIMIT 1;
```

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

```
SELECT
   (SELECT SUM(j1.expertise_level)
    FROM chef j1
```

```
    JOIN judge j ON j1.chef_id = j.chef_id
    JOIN episode_judge ej ON ej.judge_id = j.judge_id
    WHERE ej.episode_id = e.episode_id)
   +
   (SELECT SUM(j2.expertise_level)
    FROM chef j2
    JOIN assignment a ON a.chef_id = j2.chef_id
    JOIN episode_assignment ep ON ep.assignment_id = a.assignment_id
    WHERE ep.episode_id = e.episode_id) as total_expertise,
   e.episode_id
FROM episode e
ORDER BY total_expertise
LIMIT 1;
```

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

```
SELECT theme_name, count(*)
FROM recipe
GROUP BY theme_name
ORDER BY count(*) desc;
```

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

```
SELECT fg.name
FROM food_group fg
WHERE NOT EXISTS (
   SELECT 1
   FROM ingredient i
   JOIN food_group c ON i.food_group_name = c.name
   JOIN recipe_ingredient ri ON ri.ingredient = i.name
   JOIN recipe r ON r.recipe_id = ri.recipe_id
   JOIN assignment a ON r.recipe_id = a.recipe_id
   WHERE c.name = fg.name
);
```

9) Χρήστες:

Παρέχουμε την δυνατότητα στους chefs να έχουν πρόσβαση σε ορισμένα δεδομένα. Κάθε φορά μπορούμε να ορίσουμε έναν νέο χρήστη όπως στο ακόλουθο παράδειγμα, όπου ορίζουμε ως χρήστη τον chef με chef_id=105.

```
CREATE USER 'chef_105'@'localhost' IDENTIFIED BY 'password';
```

Για να γίνουν βέβαια διαθέσιμα τα δεδομένα πρέπει να δημιουργήσουμε κάποια views. Παρατίθεται ως παράδειγμα η δημιουργία του chef_view και έπειτα του recipe_view και του recipe_ingredient_view.
Chef_view:

```
CREATE VIEW chef_view AS
   SELECT *
   FROM chef
   WHERE chef_id = 105
```

Recipe_view:

```
CREATE  VIEW recipe_view AS
  SELECT *
   FROM      recipe
   WHERE
      recipe.recipe_id IN (SELECT
            assignment.recipe_id
         FROM assignment
         WHERE
            (assignment.chef_id = 105))
```

```
CREATE VIEW recipe_ingredient_view AS
   SELECT *
   FROM recipe_ingredient
   WHERE
      recipe_ingredient.recipe_id` IN (SELECT
            recipe_view.recipe_id
         FROM
            recipe_view)
```

Αντίστοιχα views φτιάχνουμε και για τα υπόλοιπα tables που συνδέονται με τις συνταγές, πχ για τον εξοπλισμό τα υλικά κλπ.
Για να δώσουμε δικαιώματα στον χρήστη να βλέπει και να επεξεργάζεται δεδομένα, καθώς και να μπορεί να εισαγάγει νέες συνταγές, χρησιμοποιούμε τον εξής κώδικα:

```
GRANT SELECT, UPDATE ON recipe_view TO chef_105;
GRANT UPDATE ON recipe TO chef_105;
```

Αντίστοιχα πράττουμε για τα υπόλοιπα views.