



ΕΞΑΜΗΝΙΑΙΑ ΕΡΓΑΣΙΑ

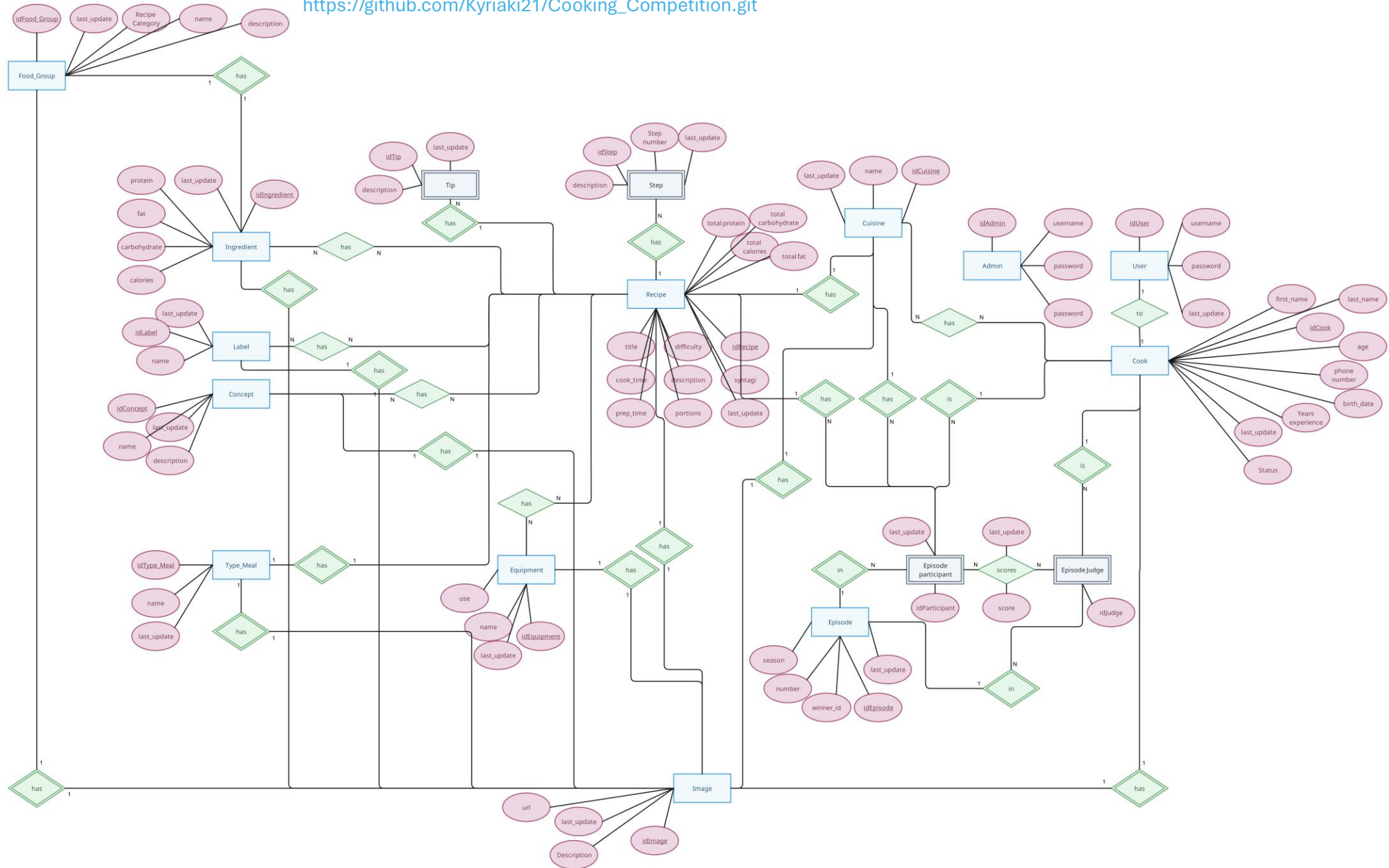
ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

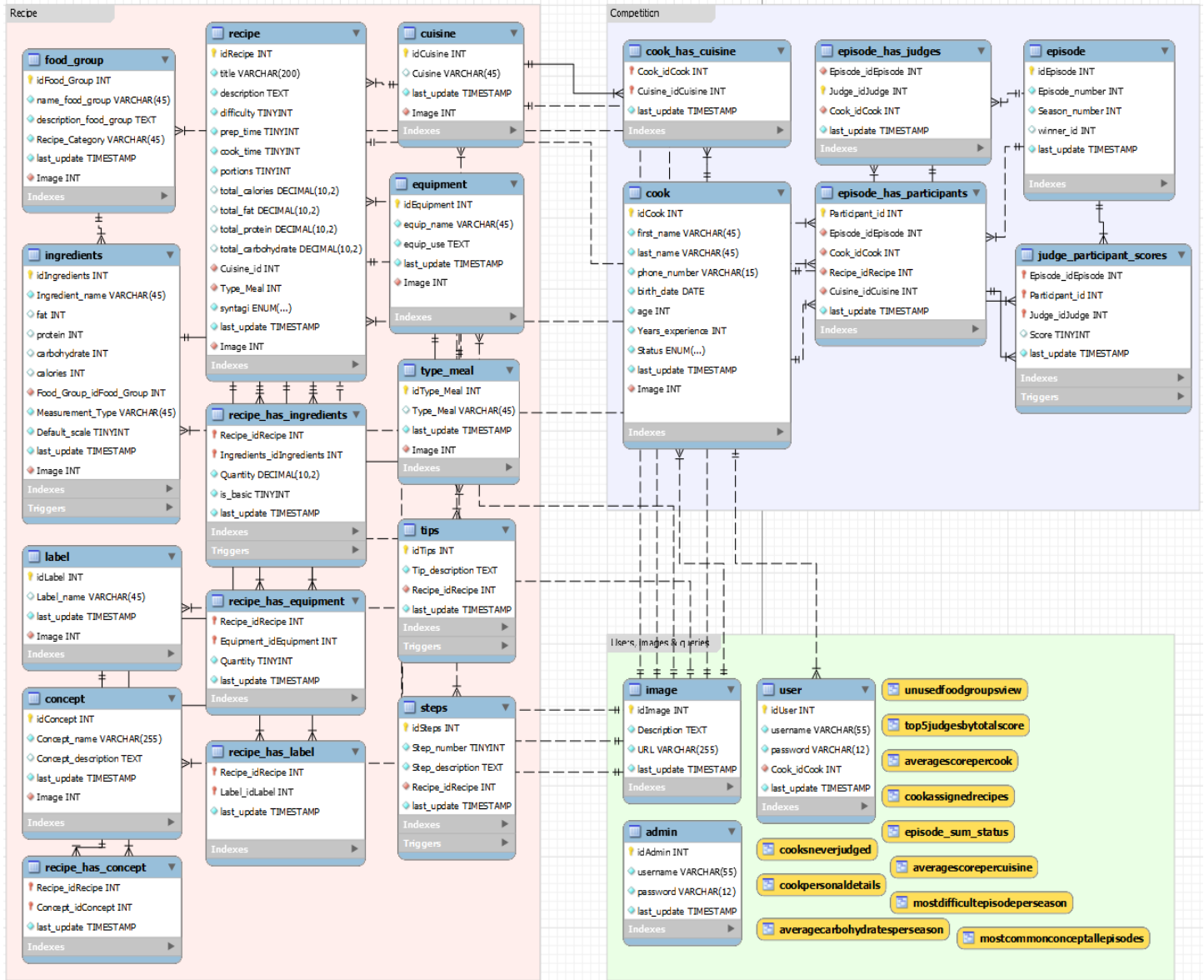
ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

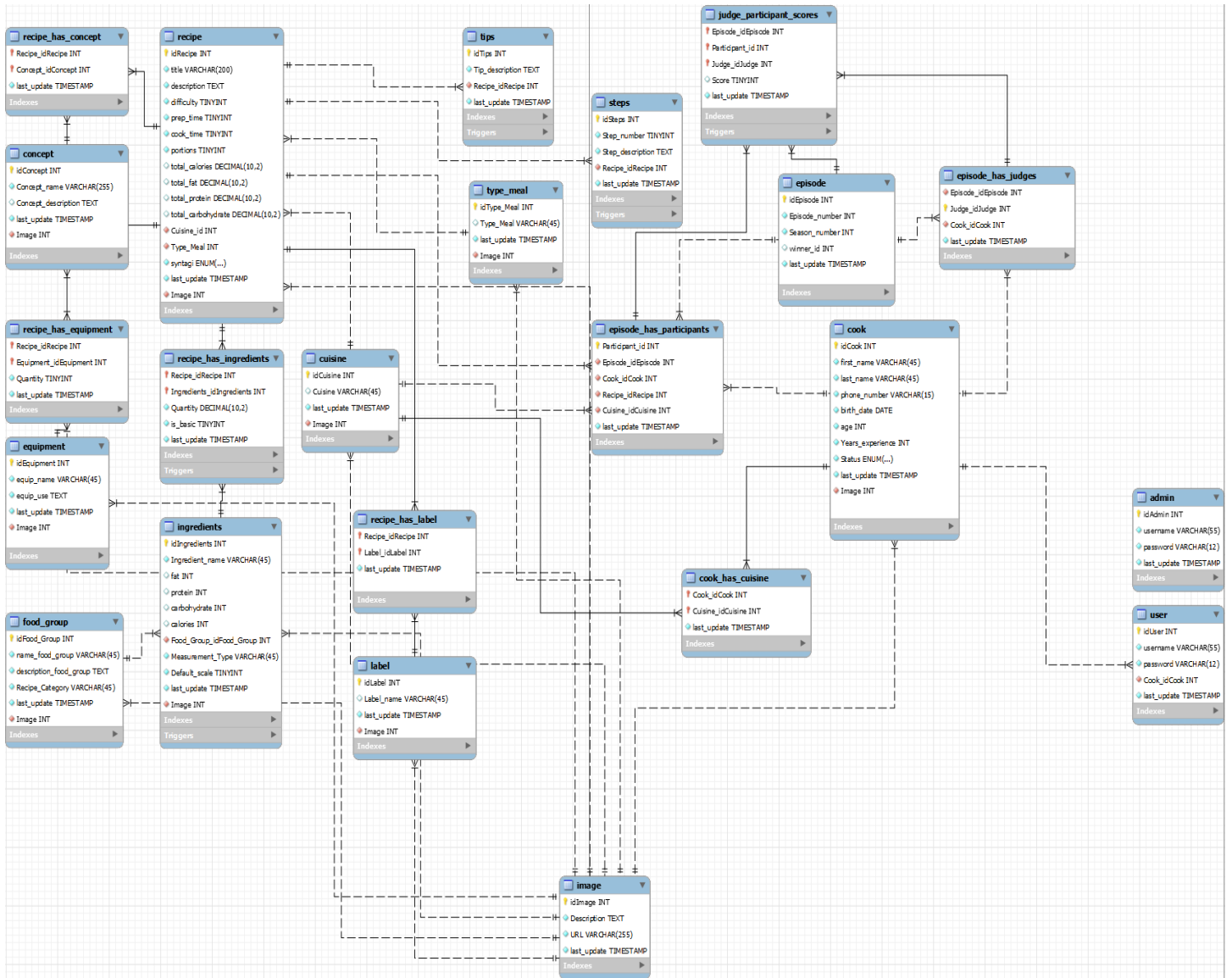
ΕΛΕΥΘΕΡΙΟΣ ΚΥΡΑΝΝΑΣ-ΚΥΡΙΑΚΗ ΜΑΖΙΩΤΗ

A.M: ge19062- ge19107

https://github.com/Kyriaki21/Cooking_Competition.git







Σύμφωνα με τα παραπάνω διαγράμματα, θα δημιουργήσουμε την βάση μας στην οποία θα έχουμε πίνακες με καθέναν από αυτόν να έχει ένα PRIMARY KEY (κύριο στοιχείο) το οποίο θα είναι ένας αύξον αριθμό που θα χρησιμοποιηθεί για να πραγματοποιήσουμε τις ενωτικές σχέσεις των στοιχείων μας που απεικονίζονται παραπάνω καθώς και το κύριο ευρετικό στοιχείο των διαδικασιών μας.

Οι ενώσεις αυτές θα έχουν ως ενωτικά στοιχεία τους τα FOREIGN KEYS (ξένα στοιχεία) τα οποία θα πρέπει να υπάρχουν παρόμοια σε όσους πίνακες ενώνουν και όπου η βάση απευθείας θα ελέγχει την ακεραιότητα τους καθώς αυτή θα μας τα παράγει. Επίσης κάθε πίνακας μας θα περιέχει και την πληροφορία της τελευταίας τροποποίησης των πληροφοριών του για λόγους κοινοχρηστικότητας της. Με τα παραπάνω σαν γνώμονα θα αρχίσουμε ορίζοντας τον πίνακα “**Image**”.

```
CREATE TABLE `Cooking_Competition`.`Image` (  
  `idImage` INT UNSIGNED NOT NULL AUTO_INCREMENT,  
  `Description` TEXT NOT NULL,  
  `URL` VARCHAR(255) NOT NULL,  
  `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  PRIMARY KEY (`idImage`)  
)  
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Ο παραπάνω πίνακας, ορίζεται πρώτος καθώς ενώνεται με σχεδόν όλους τους παρακάτω πίνακες και έχει ως Primary Key το “**idImage**” (μοναδικό για κάθε εικόνα), μια περιγραφή της εικόνας μας και ένα URL (την ηλεκτρονική διεύθυνση της φωτογραφίας μας) καθώς και την τελευταία τροποποίηση κάθε πληροφορίας μας. Επόμενος είναι ο πίνακας “**Cuisine**”

```
CREATE TABLE `Cooking_Competition`.`Cuisine` (  
  `idCuisine` INT UNSIGNED NOT NULL AUTO_INCREMENT,  
  `Cuisine` VARCHAR(45) NULL,  
  `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  `Image` INT UNSIGNED NOT NULL,  
  INDEX `fk_Cuisine_Image_idx` (`Image` ASC) ,  
  INDEX `Search_Cuisine_idx` (`Cuisine` ASC) ,  
  CONSTRAINT `fk_Cuisine_Image`  
    FOREIGN KEY (`Image`)  
    REFERENCES `Cooking_Competition`.`Image` (`idImage`)  
    ON DELETE RESTRICT  
    ON UPDATE CASCADE,  
  PRIMARY KEY (`idCuisine`)  
)  
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Ο παραπάνω πίνακας, έχει ως Primary Key το “**idCuisine**” (μοναδικό για κάθε κουζίνα), το όνομα της κουζίνας, μοναδικό για να εξασφαλίζουμε την περίπτωση διπλο-εγγραφής της, την τελευταία τροποποίηση και φυσικά την εικόνα της κουζίνας. Εδώ θέλοντας να ενώσουμε τις φωτογραφίες που θα

έχουμε στον πίνακα με τις εικόνες με τις κουζίνες μας θα ορίσουμε ως Foreign Key το “**Image**” το οποίο μας εξασφαλίζει την ακεραιότητα της πληροφορίας μας για τις εικόνες καθώς δεν γίνεται να συνδέσουμε εικόνες με τον πίνακα μας που δεν έχουν ήδη αποθηκευτεί στην βάση μας (στον πίνακα Image). Επίσης εδώ λόγω του αριθμού των αναζητήσεων που θα χρειαστεί να γίνουν και ως προς τις εικόνες (λόγω της σύνδεσης της με τον πίνακα) αλλά και ως προς το όνομα της κουζίνας, θα βάλουμε ευρετήριο και στα δύο για να επιταχύνουμε τις διαδικασίες αυτές. Επόμενος είναι ο πίνακας “**Type_Meal**”

```
• CREATE TABLE `Cooking_Competition`.`Type_Meal` (  
  `idType_Meal` INT UNSIGNED NOT NULL AUTO_INCREMENT,  
  `Type_Meal` VARCHAR(45) NULL,  
  `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  `Image` INT UNSIGNED NOT NULL,  
  INDEX `fk_Type_Meal_Image_idx` (`Image` ASC) ,  
  INDEX `Search_Type_Meal_idx` (`Type_Meal` ASC) ,  
  CONSTRAINT `fk_Type_Meal_Image`  
    FOREIGN KEY (`Image`)  
    REFERENCES `Cooking_Competition`.`Image` (`idImage`)  
    ON DELETE RESTRICT  
    ON UPDATE CASCADE,  
  PRIMARY KEY (`idType_Meal`)  
)  
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Ο παραπάνω πίνακας (παρόμοιος με τον προηγούμενο), έχει ως Primary Key το “**idType_Meal**” (μοναδικό για κάθε τύπο γεύματος), το όνομα του γεύματος, την τελευταία τροποποίηση και πάλι ως Foreign Key το “**Image**”. Επίσης έχουμε και εδώ προσθέσει ευρετήρια στις εικόνες και στο όνομα του γεύματος πάλι λόγω των πολλών αναζητήσεων για τις εικόνες και καθώς μας ζητούνται συχνά ερωτήματα για τους τύπους γεύματος. Επόμενος είναι ο πίνακας “**Food_Group**”

```
• CREATE TABLE `Cooking_Competition`.`Food_Group` (  
  `idFood_Group` INT UNSIGNED NOT NULL AUTO_INCREMENT,  
  `name_food_group` VARCHAR(45) NOT NULL,  
  `description_food_group` TEXT NOT NULL,  
  `Recipe_Category` VARCHAR(45) NOT NULL,  
  `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  `Image` INT UNSIGNED NOT NULL,  
  INDEX `fk_Food_Group_Image_idx` (`Image` ASC) ,  
  INDEX `Search_name_food_group_idx` (`name_food_group` ASC),  
  INDEX `Search_Recipe_Category_idx` (`Recipe_Category` ASC),  
  CONSTRAINT `fk_Food_Group_Image`  
    FOREIGN KEY (`Image`)  
    REFERENCES `Cooking_Competition`.`Image` (`idImage`)  
    ON DELETE RESTRICT  
    ON UPDATE CASCADE,  
  PRIMARY KEY (`idFood_Group`),  
  UNIQUE INDEX `name_food_group_UNIQUE` (`name_food_group` ASC))  
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Ο παραπάνω πίνακας, έχει ως Primary Key το “**idFood_Group**” (μοναδικό για κάθε ομάδα τροφίμων), το όνομα της ομάδας, μια σχετική περιγραφή της, την κατηγορία που αποδίδει στην συνταγή αν το βασικό της υλικό ανήκει σε αυτήν καθώς και την τελευταία τροποποίηση και τις εικόνες. Επιλέξαμε να τοποθετήσουμε στον πίνακα της ομάδας τροφίμων την κατηγορία της συνταγής μας καθώς κάθε ένα υλικό όντας βασικό σύμφωνα με την ομάδα τροφίμων στην οποία ανήκει προσδίδει άμεσα στην συνταγή μας σε μια συγκεκριμένη κατηγορία. Αυτή η σχέση είναι 1-1 και για να μην χρειαζόμαστε ολόκληρο πίνακα μπορούμε εύκολα να το τοποθετήσουμε εδώ και να είναι άμεσα συνδεδεμένο με την συνταγή μας όπως θα δούμε παρακάτω. Πάλι και εδώ θα έχουμε ως Foreign Key το “**Image**” και εδώ θα έχουμε ευρετήρια και για τις εικόνες και το όνομα της ομάδας αλλά και την κατηγορία της συνταγής μας. Επόμενος είναι ο πίνακας “**Recipe**”.

```
CREATE TABLE `Cooking_Competition`.`Recipe` (  
  `idRecipe` INT UNSIGNED NOT NULL AUTO_INCREMENT,  
  `title` VARCHAR(200) NOT NULL,  
  `description` TEXT NOT NULL,  
  `difficulty` TINYINT NOT NULL,  
  `prep_time` TINYINT NOT NULL,  
  `cook_time` TINYINT NOT NULL,  
  `portions` TINYINT NOT NULL,  
  `total_calories` DECIMAL(10,2) NULL,  
  `total_fat` DECIMAL(10,2) NULL,  
  `total_protein` DECIMAL(10,2) NULL,  
  `total_carbohydrate` DECIMAL(10,2) NULL,  
  `Cuisine_id` INT UNSIGNED NOT NULL,  
  `Type_Meal` INT UNSIGNED NOT NULL,  
  `syntagi` ENUM('cooking', 'pastry') NOT NULL,  
  `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  `Image` INT UNSIGNED NOT NULL,  
  PRIMARY KEY (`idRecipe`),  
  INDEX `fk_Recipe_Image_idx` (`Image` ASC),  
  INDEX `Search_title_idx` (`title` ASC),  
  INDEX `Search_difficulty_idx` (`difficulty` ASC),  
  CONSTRAINT `fk_Recipe_Image`  
    FOREIGN KEY (`Image`)  
    REFERENCES `Cooking_Competition`.`Image` (`idImage`)  
    ON DELETE RESTRICT  
    ON UPDATE CASCADE,  
  INDEX `fk_Recipe_Cuisine1_idx` (`Cuisine_id` ASC),  
  INDEX `fk_Recipe_Type_Meal1_idx` (`Type_Meal` ASC),  
  CONSTRAINT `fk_Recipe_Cuisine1`  
    FOREIGN KEY (`Cuisine_id`)  
    REFERENCES `Cooking_Competition`.`Cuisine` (`idCuisine`)  
    ON DELETE RESTRICT  
    ON UPDATE CASCADE,  
  CONSTRAINT `fk_Recipe_Type_Meal1`  
    FOREIGN KEY (`Type_Meal`)  
    REFERENCES `Cooking_Competition`.`Type_Meal` (`idType_Meal`)  
    ON DELETE RESTRICT  
    ON UPDATE CASCADE,  
  CONSTRAINT `range_dif` CHECK (difficulty BETWEEN 1 AND 5),  
  CONSTRAINT `prep_time_check` CHECK (prep_time >= 0),  
  CONSTRAINT `portions_check` CHECK (portions > 0),  
  CONSTRAINT `cook_time_check` CHECK (cook_time >= 0)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```


Ο παραπάνω πίνακας, έχει ως Primary Key το “**idrecipe**” (μοναδικό για κάθε συνταγή), έχει την ονομασία της συνταγής, μια σχετική περιγραφή της, τον βαθμό δυσκολίας της στον οποίο έχουμε ορίσει να δέχεται μόνο αριθμούς από το 1 μέχρι τι 5, τον χρόνο προετοιμασίας και τον χρόνο ψησίματος που έχουμε βάλει να είναι μην αρνητικοί, τις μερίδες που θα παραχθούν (όχι αρνητικές και 0), τις συνολικές θερμίδες, λίπη, πρωτεΐνη, υδατάνθρακες (όλα σε δεκαδικούς) της συνταγής τα οποία θα τα υπολογίζουμε παρακάτω μέσω των υλικών της συνταγής, την κουζίνα της συνταγής (συνδέεται με τον κωδικό της κουζίνας), τον τύπο γεύματος που ορίσαμε πιο πάνω, το τι είδους συνταγή είναι (μαγειρική/ζαχαροπλαστική) και τέλος την τελευταία τροποποίηση και τις εικόνες. Από τα παραπάνω έχουμε σαν Foreign Keys την εικόνα, την κουζίνα και τον τύπο του γεύματος για να υπάρχει η σύνδεση καθώς κάθε συνταγή έχει ένα από αυτά (σύνδεση 1 to many) στα οποία μάλιστα θα βάλουμε και ευρετήρια για την ευκολότερη αναζήτηση, ενώ θα προσθέσουμε ευρετήρια και στο όνομα της συνταγής και την δυσκολία της καθώς είναι από τα πράγματα που θα χρησιμοποιήσει κανείς για να αναζητήσει συνταγές. Αμέσως μετά θα ορίσουμε τον πίνακα “**Ingredients**”

```
● ○ CREATE TABLE `Cooking_Competition`.`Ingredients` (  
  `idIngredients` INT UNSIGNED NOT NULL AUTO_INCREMENT,  
  `Ingredient_name` VARCHAR(45) NOT NULL,  
  `fat` INT NULL,  
  `protein` INT NULL,  
  `carbohydrate` INT NULL,  
  `calories` INT NULL,  
  `Food_Group_idFood_Group` INT UNSIGNED NOT NULL,  
  `Measurement_Type` VARCHAR(45) NOT NULL,  
  `Default_scale` TINYINT NOT NULL DEFAULT 100,  
  `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  `Image` INT UNSIGNED NOT NULL,  
  INDEX `fk_Ingedient_Image_idx` (`Image` ASC) ,  
  INDEX `Search_Ingredient_name_idx` (`Ingredient_name` ASC) ,  
  CONSTRAINT `fk_Ingedient_Image`  
    FOREIGN KEY (`Image`)  
    REFERENCES `Cooking_Competition`.`Image` (`idImage`)  
    ON DELETE RESTRICT  
    ON UPDATE CASCADE,  
  PRIMARY KEY (`idIngredients`),  
  INDEX `fk_Ingredients_Food_Group_idx` (`Food_Group_idFood_Group` ASC) ,  
  CONSTRAINT `fk_Ingredients_Food_Group`  
    FOREIGN KEY (`Food_Group_idFood_Group`)  
    REFERENCES `Cooking_Competition`.`Food_Group` (`idFood_Group`)  
    ON DELETE RESTRICT  
    ON UPDATE CASCADE,  
  CONSTRAINT fat_pos_constraint CHECK (COALESCE(fat, 0) >= 0),  
  CONSTRAINT protein_pos_constraint CHECK (COALESCE(protein, 0) >= 0),  
  CONSTRAINT carbohydrate_pos_constraint CHECK (COALESCE(carbohydrate, 0) >= 0),  
  CONSTRAINT calories_pos_constraint CHECK (COALESCE(calories, 0) >= 0),  
  UNIQUE INDEX `Ingredient_name_UNIQUE` (`Ingredient_name` ASC) )  
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```


Ο παραπάνω πίνακας, έχει ως Primary Key το “**idIngredients**” (μοναδικό για κάθε υλικό), έχει το υλικό και περιέχει τα λίπη, πρωτεΐνες, υδατάνθρακες, θερμίδες του κάθε υλικού καθώς και την ομάδα τροφίμων που ανήκει το υλικό αλλά και τον τύπο μέτρησης του και την κλίμακα μέτρησης (για πόσα γραμμάρια/ml ισχύουν τα παραπάνω στοιχεία) και τέλος την τελευταία τροποποίηση και την εικόνα. Και εδώ θα έχουμε την εικόνα ως Foreign Key αλλά θα έχουμε και την ομάδα τροφίμων καθώς κάθε υλικό ανήκει σε μία ομάδα τροφίμων στα οποία φυσικά θα προσθέσουμε ευρετήρια όπως θα προσθέσουμε ευρετήριο λόγω της πιθανής συχνότητας αναζητήσεων στο όνομα του υλικού. Τέλος έχουμε ορίσει τα θεραπευτικά μας στοιχεία όλα να είναι μην αρνητικά κατά την εισαγωγή δεδομένων. Τώρα για να ενώσουμε τους πίνακες των υλικών και των συνταγών επειδή πολλές συνταγές έχουν πολλά υλικά και πολλά υλικά ανήκουν σε πολλές συνταγές θα δημιουργήσουμε και έναν ενδιάμεσο πίνακα “**Recipe_has_Ingredients**”.

```
CREATE TABLE `Cooking_Competition`.`Recipe_has_Ingredients` (  
  `Recipe_idRecipe` INT UNSIGNED NOT NULL,  
  `Ingredients_idIngredients` INT UNSIGNED NOT NULL,  
  `Quantity` DECIMAL(10,2) NOT NULL,  
  `is_basic` TINYINT NOT NULL DEFAULT 0,  
  `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  PRIMARY KEY (`Recipe_idRecipe`, `Ingredients_idIngredients`),  
  INDEX `fk_Recipe_has_Ingredients_Ingredients1_idx` (`Ingredients_idIngredients` ASC) ,  
  INDEX `fk_Recipe_has_Ingredients_Recipe1_idx` (`Recipe_idRecipe` ASC) ,  
  CONSTRAINT unique_rec_label UNIQUE (Recipe_idRecipe,Ingredients_idIngredients),  
  CONSTRAINT chk_basic CHECK (is_basic IN (0, 1)),  
  CONSTRAINT quantity_pos CHECK (Quantity > 0),  
  CONSTRAINT `fk_Recipe_has_Ingredients_Recipe1`  
    FOREIGN KEY (`Recipe_idRecipe`)  
    REFERENCES `Cooking_Competition`.`Recipe` (`idRecipe`)  
    ON DELETE RESTRICT  
    ON UPDATE CASCADE,  
  CONSTRAINT `fk_Recipe_has_Ingredients_Ingredients1`  
    FOREIGN KEY (`Ingredients_idIngredients`)  
    REFERENCES `Cooking_Competition`.`Ingredients` (`idIngredients`)  
    ON DELETE RESTRICT  
    ON UPDATE CASCADE)  
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Ο παραπάνω πίνακας, έχει ως Primary και Foreign Keys τα “**Ingredients_idIngredients**” και “**Recipe_idRecipe**” με τα οποία συνδέει τους δύο αυτούς πίνακες στον ίδιο πίνακα και εξασφαλίζει ακεραιότητα στα δεδομένα μας ορίζοντας ότι κάθε υλικό και συνταγή μπορούν να έχουν μοναδικό συνδυασμό. Επίσης έχει και την ποσότητα του υλικού στην συγκεκριμένη συνταγή και αν το υλικό είναι βασικό στην συνταγή ενώ έχει και την τελευταία τροποποίηση. Φυσικά τα μόνα ευρετήρια εδώ είναι αυτά στα Foreign Keys για την πιο γρήγορη αναζήτηση κατά την εισαγωγή δεδομένων ενώ θέλουμε και να ελέγχουμε το βασικό υλικό να είναι 0 ή 1 (εδώ δημιουργήσαμε μια αυτοσχέδια Boolean όπου 0 δεν είναι βασικό ενώ 1 είναι με αυτόματη απάντηση να είναι όχι) αλλά και οι ποσότητες να είναι θετικές.

```

● CREATE PROCEDURE CalculateRecipeNutrition()
BEGIN
    DECLARE recipe_id INT;
    DECLARE ing_calories DECIMAL(10,2);
    DECLARE ing_fat DECIMAL(10,2);
    DECLARE ing_protein DECIMAL(10,2);
    DECLARE ing_carbohydrate DECIMAL(10,2);
    DECLARE total_calories DECIMAL(10,2);
    DECLARE total_fat DECIMAL(10,2);
    DECLARE total_protein DECIMAL(10,2);
    DECLARE total_carbohydrate DECIMAL(10,2);
    -- Declare cursor variables
    DECLARE done INT DEFAULT FALSE;
    DECLARE recipe_cursor CURSOR FOR
        SELECT idRecipe FROM Cooking_Competition.Recipe;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    -- Open the cursor
    OPEN recipe_cursor;
    -- Main loop to iterate through each recipe
    main_loop: LOOP
        -- Fetch recipe id
        FETCH recipe_cursor INTO recipe_id;
        -- Exit loop if no more recipes
        IF done THEN
            LEAVE main_loop;
        END IF;
        -- Reset total nutrition values
        SET total_calories = 0;
        SET total_fat = 0;
        SET total_protein = 0;
        SET total_carbohydrate = 0;
        -- Retrieve ingredients and aggregate nutrition for the current recipe
        SELECT
            SUM(i.calories * ri.Quantity / 100) AS total_calories,
            SUM(i.fat * ri.Quantity / 100) AS total_fat,
            SUM(i.protein * ri.Quantity / 100) AS total_protein,
            SUM(i.carbohydrate * ri.Quantity / 100) AS total_carbohydrate
        INTO
            ing_calories, ing_fat, ing_protein, ing_carbohydrate
        FROM
            Cooking_Competition.Recipe_has_Ingredients ri
        INNER JOIN
            Cooking_Competition.Ingredients i ON ri.Ingredients_idIngredients = i.idIngredients
        WHERE
            ri.Recipe_idRecipe = recipe_id;
        -- Set total nutrition values
        SET total_calories = IFNULL(ing_calories, 0);
        SET total_fat = IFNULL(ing_fat, 0);
        SET total_protein = IFNULL(ing_protein, 0);
        SET total_carbohydrate = IFNULL(ing_carbohydrate, 0);
        -- Update Recipe table with total nutrition values
        UPDATE Cooking_Competition.Recipe
        SET
            total_calories = total_calories,
            total_fat = total_fat,
            total_protein = total_protein,
            total_carbohydrate = total_carbohydrate
        WHERE
            idRecipe = recipe_id;
    END LOOP;
    -- Close recipe cursor
    CLOSE recipe_cursor;
END//

```

Για τον συγκεκριμένο πίνακα μαζί με τον πίνακα των συνταγών, έχουμε ορίσει την συγκεκριμένη διαδικασία η οποία επιλέγει για ένα υλικό της συνταγής την ποσότητα του στην συνταγή τα θρεπτικά του χαρακτηριστικά και την βασική κλίμακα τους και υπολογίζει τι θρεπτική αξία προσφέρουν στην συνταγή. Αυτό το επαναλαμβάνει για κάθε υλικό της συνταγής και τέλος αποθηκεύει τα συνολικά θρεπτικά συστατικά στην συνταγή μας. Για να κάνουμε λοιπόν εμείς την βάση μας να τα υπολογίζει αυτά δυναμικά (δηλαδή με κάθε αλλαγή στα δεδομένα μας τα νούμερα στην συνταγή μας να αντιπροσωπεύουν την αλλαγή) θα χρειαστούμε και τα εξής triggers (πυροδοτητές) τα οποία ορίζουν πότε και πώς θα γίνει η επεξεργασία.

```
-- Trigger for after inserting an ingredient
DELIMITER //
```

- ```
CREATE TRIGGER AfterInsertRecipeIngredient
AFTER INSERT ON Recipe_has_Ingredients
FOR EACH ROW
BEGIN
 CALL CalculateRecipeNutrition();
END //
```

```
DELIMITER ;
```

- ```
CREATE TRIGGER AfterDeleteRecipeIngredient
AFTER DELETE ON Recipe_has_Ingredients
FOR EACH ROW
BEGIN
    CALL CalculateRecipeNutrition();
END //
```

```
DELIMITER ;
```

```
-- Trigger for updating ingredient values
DELIMITER //
```

- ```
CREATE TRIGGER BeforeUpdateIngredients
BEFORE UPDATE ON Ingredients
FOR EACH ROW
BEGIN
 IF NEW.fat != OLD.fat OR
 NEW.protein != OLD.protein OR
 NEW.carbohydrate != OLD.carbohydrate OR
 NEW.calories != OLD.calories THEN
 CALL CalculateRecipeNutrition();
 END IF;
END //
```

```
DELIMITER ;
```

Αυτό όπως βλέπουμε και παραπάνω, πρέπει να γίνεται κάθε φορά αφού προσθέσουμε ένα καινούργιο υλικό από μια συνταγή, κάθε φορά αφού διαγράψουμε ένα υλικό από μια συνταγή και κάθε φορά που τροποποιήσουμε ένα από τα θρεπτικά συστατικά κάποιου υλικού. Επίσης θέλουμε να γίνεται και κάθε φορά που τροποποιούμε πιο υλικό πηγαίνει σε ποια συνταγή και όταν αλλάζουμε τις ποσότητες τους.

```
DELIMITER //
```

```
• CREATE TRIGGER BeforeUpdateRecipe_has_Ingredients
 BEFORE UPDATE ON Recipe_has_Ingredients
 FOR EACH ROW
 BEGIN
 -- Check if any of the nutritional values are being updated
 IF NEW.Ingredients_idIngredients != OLD.Ingredients_idIngredients OR
 NEW.Quantity != OLD.Quantity OR
 NEW.Recipe_idRecipe != OLD.Recipe_idRecipe THEN
 CALL CalculateRecipeNutrition();
 END IF;
 END //

DELIMITER ;
```

Επόμενο πίνακα για την βάση θα ορίσουμε τον πίνακα “**Tips**”.

```
• CREATE TABLE `Cooking_Competition`.`Tips` (
 `idTips` INT UNSIGNED NOT NULL AUTO_INCREMENT,
 `Tip_description` TEXT NOT NULL,
 `Recipe_idRecipe` INT UNSIGNED NOT NULL,
 `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
 PRIMARY KEY (`idTips`),
 INDEX `fk_Tips_Recipe1_idx` (`Recipe_idRecipe` ASC) ,
 CONSTRAINT `fk_Tips_Recipe1`
 FOREIGN KEY (`Recipe_idRecipe`)
 REFERENCES `Cooking_Competition`.`Recipe` (`idRecipe`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Ο παραπάνω πίνακας, έχει ως Primary Key το “**idTips**” (μοναδικό για κάθε συμβουλή), μια περιγραφή της συμβουλής, την συνταγή που ανήκει η συμβουλή και την τελευταία τροποποίηση. Ως Foreign Key έχουμε φυσικά την συνταγή με την οποία ενώνεται κάθε συμβουλή μοναδικά με μια συνταγή στην οποία βάλαμε και ένα ευρετήριο για ευκολότερες αναζητήσεις. Για τις συμβουλές έχουν περιορισμό μέχρι 3 για κάθε συνταγή το οποίο θα τον εκτελέσουμε μέσω του trigger:

```
-- Trigger to limit the number of tips per recipe checked
```

```
DELIMITER //
```

```
• CREATE TRIGGER CheckTipsCount
 BEFORE INSERT ON Tips
 FOR EACH ROW
 BEGIN
 DECLARE tips_count INT;

 -- Count the number of tips for the given recipe
 SELECT COUNT(*) INTO tips_count
 FROM Tips
 WHERE Recipe_idRecipe = NEW.Recipe_idRecipe;

 -- If the count exceeds 3, raise an error
 IF tips_count >= 3 THEN
 SIGNAL SQLSTATE '45000'
 SET MESSAGE_TEXT = 'A recipe cannot have more than 3 tips';
 END IF;
 END//
```

```
DELIMITER ;
```

Η παραπάνω διαδικασία γίνεται κάθε φορά πριν εισάγουμε μια καινούργια συμβουλή και επιλέγει τις ήδη υπάρχον συμβουλές για την συνταγή στην οποία προσπαθούμε να εισάγουμε την καινούργια συμβουλή και τις μετράει. Αν είναι 3 και πάνω τότε δεν επιτρέπει την εισαγωγή.

Επόμενο πίνακα για την βάση θα ορίσουμε τον πίνακα “**Steps**”.

```
• CREATE TABLE `Cooking_Competition`.`Steps` (
 `idSteps` INT UNSIGNED NOT NULL AUTO_INCREMENT,
 `Step_number` TINYINT NOT NULL,
 `Step_description` TEXT NOT NULL,
 `Recipe_idRecipe` INT UNSIGNED NOT NULL,
 `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
 PRIMARY KEY (`idSteps`),
 INDEX `fk_Steps_Recipe1_idx` (`Recipe_idRecipe` ASC) ,
 INDEX `Search_Step_number_idx` (`Step_number` ASC) ,
 CONSTRAINT `fk_Steps_Recipe1`
 FOREIGN KEY (`Recipe_idRecipe`)
 REFERENCES `Cooking_Competition`.`Recipe` (`idRecipe`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Ο παραπάνω πίνακας, έχει ως Primary Key το “**idSteps**” (μοναδικό για κάθε βήμα της συνταγής), τον αύξον αριθμό του βήματος (για την συγκεκριμένη συνταγή καθώς θέλουμε να μπορούμε να τα εμφανίζουμε σειριακά με συγκεκριμένη δοσμένη σειρά), το κείμενο του βήματος, την συνταγή φυσικά και την τελευταία τροποποίηση. Ως Foreign Key έχουμε την συνταγή με την οποία ενώνεται κάθε βήμα μοναδικά με μια συνταγή στην οποία βάλαμε και ένα ευρετήριο για ευκολότερες αναζητήσεις. Για τα βήματα έχουμε όπως είπαμε την ανάγκη να είναι σειριακά όποτε ζητάμε από τον χρήστη να τα προσθέτει σειριακά. Αυτό επιτυγχάνεται αναγκάζοντας τον να προσθέτει τα βήματα με την σειρά χωρίς παραλείψεις (δηλαδή αν υπάρχουν ήδη 3 βήματα αναγκαστικά αυτά είναι τα 1 2 3 και ο χρήστης μπορεί να προσθέσει μόνο το 4 και κανένα άλλο).

```
--
-- Trigger to ensure steps are inserted in order
--

DELIMITER //

• CREATE TRIGGER BeforeInsertStep
 BEFORE INSERT ON Steps
 FOR EACH ROW
 BEGIN
 DECLARE prevStep TINYINT;

 -- Find the step number of the previous step for the given recipe
 SELECT Step_number
 INTO prevStep
 FROM Steps
 WHERE Recipe_idRecipe = NEW.Recipe_idRecipe
 ORDER BY Step_number DESC
 LIMIT 1;

 -- If the previous step exists and the new step number is not consecutive, raise an error
 IF prevStep IS NOT NULL AND NEW.Step_number != prevStep + 1 THEN
 SIGNAL SQLSTATE '45000'
 SET MESSAGE_TEXT = 'Previous step does not exist or step numbers are not consecutive for this recipe. Please insert the steps for this recipe in the correct order.';
 END IF;
 END //

DELIMITER ;
```

Το trigger που το επιτυγχάνει αυτό ελέγχει πριν κάθε εισαγωγή στον πίνακα μας πιο είναι το προηγούμενο βήμα για αυτή την συνταγή και αν τα βήματα που προσπαθούμε να προσθέσουμε δεν είναι το επόμενο του δεν μας το επιτρέπει.

Επόμενο πίνακα για την βάση θα ορίσουμε τον πίνακα “**Equipment**”.

```
• CREATE TABLE `Cooking_Competition`.`Equipment` (
 `idEquipment` INT UNSIGNED NOT NULL AUTO_INCREMENT,
 `equip_name` VARCHAR(45) NOT NULL,
 `equip_use` TEXT NOT NULL,
 `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
 `Image` INT UNSIGNED NOT NULL,
 INDEX `fk_Equipment_Image_idx` (`Image` ASC) ,
 INDEX `Search_equip_name_idx` (`equip_name` ASC) ,
 CONSTRAINT `fk_Equipment_Image`
 FOREIGN KEY (`Image`)
 REFERENCES `Cooking_Competition`.`Image` (`idImage`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE,
 PRIMARY KEY (`idEquipment`),
 UNIQUE INDEX `equip_name_UNIQUE` (`equip_name` ASC))
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Ο παραπάνω πίνακας, έχει ως Primary Key το “***idEquipment***” (μοναδικό για κάθε βήμα της εξάρτημα), το όνομα του εξαρτήματος, την χρησιμότητά του, την τελευταία τροποποίηση και την εικόνα. Ως Foreign Key έχουμε μόνο την εικόνα και αντίστοιχο ευρετήριο για αυτή και προσθέσαμε ένα ευρετήριο για το όνομα του εξαρτήματος λόγω πιθανού όγκου αναζητήσεων. Επίσης θέλουμε και το κάθε υλικό να είναι μοναδικό για να αποφύγουμε τα διπλότυπα.

Επόμενο πίνακα για την βάση θα ορίσουμε τον πίνακα “***Recipe\_has\_Equipment***” ο οποίος είναι ενδιάμεσος πίνακας που συνδέει τις συνταγές με τον εξοπλισμό τους όπου μια συνταγή μπορεί να έχει διάφορα εξαρτήματα και κάθε εξάρτημα μπορεί να ανήκει σε πολλές συνταγές.

```
CREATE TABLE `Cooking_Competition`.`Recipe_has_Equipment` (
 `Recipe_idRecipe` INT UNSIGNED NOT NULL,
 `Equipment_idEquipment` INT UNSIGNED NOT NULL,
 `Quantity` TINYINT NOT NULL,
 `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
 PRIMARY KEY (`Recipe_idRecipe`, `Equipment_idEquipment`),
 INDEX `fk_Recipe_has_Equipment_Equipment1_idx` (`Equipment_idEquipment` ASC),
 INDEX `fk_Recipe_has_Equipment_Recipe1_idx` (`Recipe_idRecipe` ASC),
 CONSTRAINT `unique_rec equip` UNIQUE (`Recipe_idRecipe`, `Equipment_idEquipment`),
 CONSTRAINT `fk_Recipe_has_Equipment_Recipe1`
 FOREIGN KEY (`Recipe_idRecipe`)
 REFERENCES `Cooking_Competition`.`Recipe` (`idRecipe`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE,
 CONSTRAINT `fk_Recipe_has_Equipment_Equipment1`
 FOREIGN KEY (`Equipment_idEquipment`)
 REFERENCES `Cooking_Competition`.`Equipment` (`idEquipment`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Ο παραπάνω πίνακας, έχει ως Primary και Foreign Keys τα “***Recipe\_idRecipe***” και “***Equipment\_idEquipment***” τα οποία φυσικά θα έχουν το δικό τους ευρετήριο και χρησιμοποιούνται για την μοναδικότητα κάθε συνδυασμού συνταγής και εξοπλισμού. Ακόμη, έχει και την ποσότητα των εξαρτημάτων που χρειαζόμαστε για την εκάστοτε συνταγή και την τελευταία τροποποίηση.

Επόμενο πίνακα για την βάση θα ορίσουμε τον πίνακα “***Concept***”.



- ```

CREATE TABLE `Cooking_Competition`.`Concept` (
  `idConcept` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `Concept_name` VARCHAR(255) NOT NULL,
  `Concept_description` TEXT DEFAULT NULL,
  `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `Image` INT UNSIGNED NOT NULL,
  INDEX `fk_Concept_Image_idx` (`Image` ASC) ,
  UNIQUE INDEX `Concept_name_UNIQUE` (`Concept_name` ASC),
  CONSTRAINT `fk_Concept_Image`
    FOREIGN KEY (`Image`)
      REFERENCES `Cooking_Competition`.`Image` (`idImage`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  PRIMARY KEY (`idConcept`))
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

Ο παραπάνω πίνακας, έχει ως Primary Key το “**idConcept**” (μοναδικό για κάθε θεματική ενότητα), το όνομα της , μια περιγραφή της, την τελευταία τροποποίηση και την εικόνα. Ως Foreign Key έχουμε μόνο την εικόνα και αντίστοιχο ευρετήριο για αυτή και προσθέσαμε ένα ευρετήριο για το όνομα της θεματικής ενότητας λόγω πιθανού όγκου αναζητήσεων ενώ θέλουμε και το όνομα κάθε θεματικής ενότητας να είναι μοναδικό για αποφυγή διπλότυπων.

Επόμενο πίνακα για την βάση θα ορίσουμε τον πίνακα “**Recipe_has_Concept**” ο οποίος είναι ενδιάμεσος πίνακας που συνδέει τις συνταγές με τις θεματικές ενότητες τους όπου μια συνταγή μπορεί να έχει διάφορες θεματικές ενότητες και κάθε θεματική ενότητα μπορεί να ανήκει σε πολλές συνταγές.

- ```

CREATE TABLE `Cooking_Competition`.`Recipe_has_Concept` (
 `Recipe_idRecipe` INT UNSIGNED NOT NULL,
 `Concept_idConcept` INT UNSIGNED NOT NULL,
 `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
 PRIMARY KEY (`Recipe_idRecipe`, `Concept_idConcept`),
 INDEX `fk_Recipe_has_Concept_Concept1_idx` (`Concept_idConcept` ASC) ,
 INDEX `fk_Recipe_has_Concept_Recipe1_idx` (`Recipe_idRecipe` ASC) ,
 CONSTRAINT unique_rec_concept UNIQUE (Recipe_idRecipe,Concept_idConcept),
 CONSTRAINT `fk_Recipe_has_Concept_Recipe1`
 FOREIGN KEY (`Recipe_idRecipe`)
 REFERENCES `Cooking_Competition`.`Recipe` (`idRecipe`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE,
 CONSTRAINT `fk_Recipe_has_Concept_Concept1`
 FOREIGN KEY (`Concept_idConcept`)
 REFERENCES `Cooking_Competition`.`Concept` (`idConcept`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

Ο παραπάνω πίνακας, έχει ως Primary και Foreign Keys τα “**Recipe\_idRecipe**” και “**Concept\_idConcept**” τα οποία φυσικά θα έχουν το δικό τους ευρετήριο και χρησιμοποιούνται για την μοναδικότητα κάθε συνδυασμού συνταγής και θεματικής ενότητας αλλά και την τελευταία τροποποίηση. Όπως φαίνεται ο πίνακας είναι καθαρά ενδιάμεση ένωση των άλλων δύο χωρίς περαιτέρω πληροφορίες.

Επόμενο πίνακα για την βάση θα ορίσουμε τον πίνακα “**Label**”.

```
CREATE TABLE `Cooking_Competition`.`Label` (
 `idLabel` INT UNSIGNED NOT NULL AUTO_INCREMENT,
 `Label_name` VARCHAR(45) NULL,
 `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
 `Image` INT UNSIGNED NOT NULL,
 PRIMARY KEY (`idLabel`),
 UNIQUE INDEX `Label_name_UNIQUE` (`Label_name` ASC),
 INDEX `fk_Label_Image_idx` (`Image` ASC),
 CONSTRAINT `fk_Label_Image`
 FOREIGN KEY (`Image`)
 REFERENCES `Cooking_Competition`.`Image` (`idImage`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Ο παραπάνω πίνακας, έχει ως Primary Key το “**idLabel**” (μοναδικό για κάθε ετικέτα της συνταγής) και το όνομα της αλλά και την τελευταία τροποποίηση και την εικόνα. Ως Foreign Key έχουμε μόνο την εικόνα και αντίστοιχο ευρετήριο για αυτή και προσθέσαμε ένα ευρετήριο για το όνομα της ετικέτας λόγω πιθανού όγκου αναζητήσεων.

Επόμενο πίνακα για την βάση θα ορίσουμε τον πίνακα “**Recipe\_has\_Label**” ο οποίος είναι ενδιάμεσος πίνακας που συνδέει τις συνταγές με τις ετικέτες τους όπου μια συνταγή μπορεί να έχει διάφορες ετικέτες και κάθε ετικέτα μπορεί να ανήκει σε πολλές συνταγές.

```
CREATE TABLE `Cooking_Competition`.`Recipe_has_Label` (
 `Recipe_idRecipe` INT UNSIGNED NOT NULL,
 `Label_idLabel` INT UNSIGNED NOT NULL,
 `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
 PRIMARY KEY (`Recipe_idRecipe`, `Label_idLabel`),
 INDEX `fk_Recipe_has_Label_Label1_idx` (`Label_idLabel` ASC),
 INDEX `fk_Recipe_has_Label_Recipe1_idx` (`Recipe_idRecipe` ASC),
 CONSTRAINT `fk_Recipe_has_Label_Recipe1`
 FOREIGN KEY (`Recipe_idRecipe`)
 REFERENCES `Cooking_Competition`.`Recipe` (`idRecipe`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE,
 CONSTRAINT `fk_Recipe_has_Label_Label1`
 FOREIGN KEY (`Label_idLabel`)
 REFERENCES `Cooking_Competition`.`Label` (`idLabel`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Ο παραπάνω πίνακας, έχει ως Primary και Foreign Keys τα “**Recipe\_idRecipe**” και “**Label\_idLabel**” τα οποία φυσικά θα έχουν το δικό τους ευρετήριο και χρησιμοποιούνται για την μοναδικότητα κάθε συνδυασμού συνταγής και ετικέτας αλλά και την τελευταία τροποποίηση. Όπως φαίνεται ο πίνακας είναι καθαρά ενδιάμεση ένωση των άλλων δύο χωρίς περαιτέρω πληροφορίες.

Έχοντας πλέον τελειώσει για τα δεδομένα μας με τις συνταγές, προχωράμε στα δεδομένα του διαγωνισμού όπου θα αρχίσουμε φυσικά με τον πίνακα “**Cook**”.

```
• CREATE TABLE `Cooking_Competition`.`Cook` (
 `idCook` INT UNSIGNED NOT NULL AUTO_INCREMENT,
 `first_name` VARCHAR(45) NOT NULL,
 `last_name` VARCHAR(45) NOT NULL,
 `phone_number` VARCHAR(15) NOT NULL,
 `birth_date` DATE NOT NULL,
 `age` INT NOT NULL,
 `Years_experience` INT NOT NULL,
 `Status` ENUM('C cook', 'B cook', 'A cook', 'assistant head Chef', 'Chef') NOT NULL,
 `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
 `Image` INT UNSIGNED NOT NULL,
 INDEX `Search_First_name_idx` (`first_name` ASC),
 INDEX `Search_Last_name_idx` (`last_name` ASC),
 INDEX `Search_Years_experience_idx` (`Years_experience` ASC),
 INDEX `Search_Status_idx` (`Status` ASC),
 INDEX `fk_Cook_Image_idx` (`Image` ASC) ,
 CONSTRAINT `fk_Cook_Image`
 FOREIGN KEY (`Image`)
 REFERENCES `Cooking_Competition`.`Image` (`idImage`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE,
 PRIMARY KEY (`idCook`))
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Ο παραπάνω πίνακας, έχει ως Primary Key το “**idCook**” (μοναδικό για κάθε μάγειρα), το όνομα και το επώνυμο του, τον αριθμό του τηλεφώνου του, την ημερομηνία γέννησής του, την ηλικία του (όπου είναι θετική), τα χρόνια εξειδίκευσης τους (θετικά), τον τίτλο του (Γ μάγειρας, Β μάγειρας, Α μάγειρας, βοηθός σεφ, σεφ) και την τελευταία τροποποίηση και την εικόνα. Το Foreign Key μας εδώ είναι μόνο η εικόνα στην οποία βάλαμε και ένα ευρετήριο ενώ έχουμε και ευρετήρια στο όνομα, το επώνυμο και τον τίτλο του καθώς εκεί αναμένουμε τις περισσότερες αναζητήσεις από τους χρήστες μας.

Επόμενο πίνακα για την βάση θα ορίσουμε τον πίνακα “**Cook\_has\_Cuisine**” ο οποίος είναι ενδιάμεσος πίνακας που συνδέει τους μάγειρες με τις κουζίνες στις οποίες έχουν εξειδίκευση όπου ένας μάγειρας μπορεί να έχει διάφορες εξειδικεύσεις σε κουζίνες και κάθε εξειδίκευση μπορεί να ανήκει σε πολλούς μάγειρες.

```

• CREATE TABLE `Cooking_Competition`.`Cook_has_Cuisine` (
 `Cook_idCook` INT UNSIGNED NOT NULL,
 `Cuisine_idCuisine` INT UNSIGNED NOT NULL,
 `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
 PRIMARY KEY (`Cook_idCook`, `Cuisine_idCuisine`),
 INDEX `fk_Cook_Cuisine_Cuisine1_idx` (`Cuisine_idCuisine` ASC),
 INDEX `fk_Cook_Cuisine_Cook1_idx` (`Cook_idCook` ASC),
 CONSTRAINT `unique_rec_concept` UNIQUE (`Cook_idCook`, `Cuisine_idCuisine`),
 CONSTRAINT `fk_Cook_Cuisine_Cook1`
 FOREIGN KEY (`Cook_idCook`)
 REFERENCES `Cooking_Competition`.`Cook` (`idCook`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE,
 CONSTRAINT `fk_Cook_Cuisine_Cuisine1`
 FOREIGN KEY (`Cuisine_idCuisine`)
 REFERENCES `Cooking_Competition`.`Cuisine` (`idCuisine`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

Ο παραπάνω πίνακας, έχει ως Primary και Foreign Keys τα “**Cook\_idCook**” και “**Cuisine\_idCuisine**” τα οποία φυσικά θα έχουν το δικό τους ευρετήριο και χρησιμοποιούνται για την μοναδικότητα κάθε συνδυασμού μάγειρα και κουζίνα στην οποία εξειδικεύεται αλλά και την τελευταία τροποποίηση. Όπως φαίνεται ο πίνακας είναι καθαρά ενδιάμεση ένωση των άλλων δύο χωρίς περαιτέρω πληροφορίες.

Επόμενο πίνακα για την βάση θα ορίσουμε τον πίνακα “**Episode**”.

```

• CREATE TABLE `Cooking_Competition`.`Episode` (
 `idEpisode` INT UNSIGNED NOT NULL AUTO_INCREMENT,
 `Episode_number` INT NOT NULL,
 `Season_number` INT UNSIGNED NOT NULL,
 `winner_id` INT UNSIGNED NULL,
 `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
 PRIMARY KEY (`idEpisode`),
 INDEX `Search_Episode_number_idx` (`Episode_number` ASC),
 INDEX `Search_Season_number_idx` (`Season_number` ASC),
 INDEX `Search_Cook_Winner_idx` (`winner_id` ASC)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

Ο παραπάνω πίνακας, έχει ως Primary Key το “**idEpisode**” (μοναδικό για κάθε επεισόδιο), τον αριθμό του επεισοδίου, την σεζόν στην οποία ανήκει, τον νικητή του επεισοδίου και την τελευταία τροποποίηση. Ο νικητής του επεισοδίου τον έχουμε βάλει να μπορεί να είναι null καθώς πρέπει πρώτα να αποθηκεύσουμε τις βαθμολογίες του επεισοδίου για να τον υπολογίσουμε και μετά να τον αποθηκεύσουμε. Εδώ δεν έχουμε Foreign Keys καθώς θα χρησιμοποιήσουμε αυτόν τον πίνακα σαν κύριο και θα τον ενώσουμε μετά

με τους άλλους ενώ έχουμε ευρετήρια και για τον αριθμό του επεισοδίου και της σεζόν αλλά και τον νικητή καθώς και για τους υπολογισμούς που θα χρειαστούμε αλλά και από τους χρήστες θα υπάρχει πληθώρα αναζητήσεων.

Επόμενο πίνακα για την βάση θα ορίσουμε τον πίνακα “***Episode\_has\_Participants***”.

```
CREATE TABLE `Cooking_Competition`.`Episode_has_Participants` (
 `Participant_id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
 `Episode_idEpisode` INT UNSIGNED NOT NULL,
 `Cook_idCook` INT UNSIGNED NOT NULL,
 `Recipe_idRecipe` INT UNSIGNED NOT NULL,
 `Cuisine_idCuisine` INT UNSIGNED NOT NULL,
 `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
 PRIMARY KEY (`Participant_id`),
 INDEX `fk_Episode_Participant_Episode1_idx` (`Episode_idEpisode` ASC),
 INDEX `fk_Episode_Participant_Cook1_idx` (`Cook_idCook` ASC),
 INDEX `fk_Episode_Participant_Recipe1_idx` (`Recipe_idRecipe` ASC),
 INDEX `fk_Episode_Participant_Cuisine1_idx` (`Cuisine_idCuisine` ASC),
 UNIQUE INDEX `unique_Episode_Cook` (`Episode_idEpisode`, `Cook_idCook`),
 UNIQUE INDEX `unique_Episode_Cuisine` (`Episode_idEpisode`, `Cuisine_idCuisine`),
 UNIQUE INDEX `unique_Episode_Recipe` (`Episode_idEpisode`, `Recipe_idRecipe`),
 CONSTRAINT `fk_Episode_Participant_Episode1`
 FOREIGN KEY (`Episode_idEpisode`)
 REFERENCES `Cooking_Competition`.`Episode` (`idEpisode`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE,
 CONSTRAINT `fk_Episode_Participant_Cook1`
 FOREIGN KEY (`Cook_idCook`)
 REFERENCES `Cooking_Competition`.`Cook` (`idCook`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE,
 CONSTRAINT `fk_Episode_Participant_Recipe1`
 FOREIGN KEY (`Recipe_idRecipe`)
 REFERENCES `Cooking_Competition`.`Recipe` (`idRecipe`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE,
 CONSTRAINT `fk_Episode_Participant_Cuisine1`
 FOREIGN KEY (`Cuisine_idCuisine`)
 REFERENCES `Cooking_Competition`.`Cuisine` (`idCuisine`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Ο παραπάνω πίνακας, έχει ως Primary K

ey το “**Participant\_id**” μοναδικό για κάθε διαγωνιζόμενο στον οποίο διαγωνιζόμενο έχουμε επίσης βάλει το επεισόδιο στο οποίο διαγωνίζεται, το id του μάγειρα που πλέον είναι διαγωνιζόμενος, το id της συνταγής, το id της κουζίνας και την τελευταία τροποποίηση. Φυσικά τα “**Episode\_idEpisode**” “**Cook\_idCook**” “**Recipe\_idRecipe**” “**Cuisine\_idCuisine**” είναι Foreign Keys καθώς ενώνουν τους διαγωνιζόμενους με τα στοιχεία τους από τους αντίστοιχους πίνακες όπου φυσικά σε καθένα από αυτά έχουμε προσθέσει ένα ευρετήριο για ευκολότερη αναζήτηση. Επίσης θέλουμε να έχουμε μοναδικό συνδυασμό επεισοδίου/μάγειρα/κουζίνας/συνταγής το οποίο το επιτυγχάνουμε με το να έχουμε μοναδικό συνδυασμό του επεισοδίου με καθένα από αυτά άρα αυτόματα έχουμε και τις ενδιαμέσες σχέσεις μοναδικότητας. Αυτά το ομαδοποιήσαμε γιατί κάθε id διαγωνιζόμενου (κάθε φορά που κάποιος μάγειρας παίζει στον διαγωνισμό παίζει με διαφορετικό id) έχει μοναδικό επεισόδιο, μάγειρα, κουζίνα, συνταγή.

Ως αφορά τους κριτές επειδή κάθε επεισόδιο έχει 3 κριτές που επιλέγονται από τους γενικούς μάγειρες θα έχουμε τον πίνακα “**Episode\_has\_Judges**”

```
CREATE TABLE `Cooking_Competition`.`Episode_has_Judges` (
 `Episode_idEpisode` INT UNSIGNED NOT NULL,
 `Judge_idJudge` INT UNSIGNED NOT NULL AUTO_INCREMENT,
 `Cook_idCook` INT UNSIGNED NOT NULL,
 `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
 PRIMARY KEY (`Judge_idJudge`),
 INDEX `fk_Episode_Judge_Episode1_idx` (`Episode_idEpisode` ASC),
 INDEX `fk_Episode_Judge_Cook1_idx` (`Cook_idCook` ASC),
 CONSTRAINT `fk_Episode_Judge_Episode1`
 FOREIGN KEY (`Episode_idEpisode`)
 REFERENCES `Cooking_Competition`.`Episode` (`idEpisode`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE,
 CONSTRAINT `fk_Episode_Judge_Cook1`
 FOREIGN KEY (`Cook_idCook`)
 REFERENCES `Cooking_Competition`.`Cook` (`idCook`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Ο παραπάνω πίνακας, έχει ως Primary Keys το “**Judge\_idJudge**” μοναδικό για κάθε κριτή στον οποίο έχουμε επίσης βάλει το επεισόδιο στο οποίο κρίνει, το id του μάγειρα που θα είναι κριτής και την τελευταία τροποποίηση. Φυσικά τα “**Episode\_idEpisode**” “**Cook\_idCook**” είναι Foreign Keys καθώς ενώνουν τους κριτές με το επεισόδιο που θα κρίνουν και με το ποιος μάγειρας θα είναι τελικά ο κριτής με το ίδιο τρόπο όπως πριν.

Τέλος θέλουμε και έναν πίνακα για να αποθηκεύουμε τις βαθμολογίες που θα είναι ο “**Judge\_Participant\_Scores**”

```

• CREATE TABLE `Cooking_Competition`.`Judge_Participant_Scores` (
 `Episode_idEpisode` INT UNSIGNED NOT NULL,
 `Participant_id` INT UNSIGNED NOT NULL,
 `Judge_idJudge` INT UNSIGNED NOT NULL,
 `Score` TINYINT NULL CHECK (Score BETWEEN 1 AND 5),
 `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
 PRIMARY KEY (`Episode_idEpisode`, `Judge_idJudge`, `Participant_id`),
 INDEX `fk_Judge_Participant_Scores_Episode1_idx` (`Episode_idEpisode` ASC),
 INDEX `fk_Judge_Participant_Scores_Judge1_idx` (`Judge_idJudge` ASC),
 INDEX `fk_Judge_Participant_Scores_Participant1_idx` (`Participant_id` ASC),
 CONSTRAINT `fk_Judge_Participant_Scores_Episode1`
 FOREIGN KEY (`Episode_idEpisode`)
 REFERENCES `Cooking_Competition`.`Episode` (`idEpisode`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE,
 CONSTRAINT `fk_Judge_Participant_Scores_Judge1`
 FOREIGN KEY (`Judge_idJudge`)
 REFERENCES `Cooking_Competition`.`Episode_has_Judges` (`Judge_idJudge`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE,
 CONSTRAINT `fk_Judge_Participant_Scores_Participant1`
 FOREIGN KEY (`Participant_id`)
 REFERENCES `Cooking_Competition`.`Episode_has_Participants` (`Participant_id`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

Ο παραπάνω πίνακας, έχει ως Primary και Foreign Keys τα “**Episode\_idEpisode**” και “**Participant\_id**” και “**Judge\_idJudge**” όπου ενώνει με μοναδικό τρόπο και τα τρία μεταξύ τους, δηλαδή στο επεισόδιο ο κριτής έκρινε τον διαγωνιζόμενο. Επίσης πρέπει να βάλουμε και την βαθμολογία αλλά και φυσικά την τελευταία τροποποίηση. Για τις βαθμολογίες, έχουμε βάλει περιορισμό να είναι από 1 έως 5 όπως έχει ζητηθεί. Αυτός ο πίνακας εξαρτάται άμεσα από αυτούς για τα επεισόδια, διαγωνιζόμενους και κριτές.

Φυσικά ο νικητής δεν εισάγεται πριν το πέρας το διαγωνισμού για αυτό και του επιτρέπουμε να τον αφήσει κενό ενώ με την παρακάτω διαδικασία τον επιλέγουμε εμείς για αυτόν.



DELIMITER //

```
• CREATE PROCEDURE CalculateEpisodeWinner(IN inputEpisodeID INT UNSIGNED)
BEGIN
 DECLARE winner_cook_id INT UNSIGNED;

 -- Create a temporary table to hold the possible winners
 CREATE TEMPORARY TABLE TempWinners AS
 SELECT
 ep.Cook_idCook,
 SUM(jps.Score) AS TotalScore,
 c.Status
 FROM
 Judge_Participant_Scores jps
 INNER JOIN
 Episode_has_Participants ep ON jps.Participant_id = ep.Participant_id
 INNER JOIN
 Cook c ON ep.Cook_idCook = c.idCook
 WHERE
 jps.Episode_idEpisode = inputEpisodeID
 GROUP BY
 ep.Cook_idCook;

 -- Determine the highest total score
 SET @max_score = (SELECT MAX(TotalScore) FROM TempWinners);

 -- Create another temporary table to hold cooks with the highest score
 CREATE TEMPORARY TABLE TopScorers AS
 SELECT *
 FROM TempWinners
 WHERE TotalScore = @max_score;

 -- Determine the highest status among top scorers
 SET @max_status = (SELECT MIN(FIELD(Status, 'Chef', 'assistant head Chef', 'A cook', 'B cook', 'C cook')) FROM TopScorers);

 -- Create another temporary table to hold cooks with the highest status
 CREATE TEMPORARY TABLE FinalWinners AS
 SELECT *
 FROM TopScorers
 WHERE FIELD(Status, 'Chef', 'assistant head Chef', 'A cook', 'B cook', 'C cook') = @max_status;

 -- Select a random winner from the final winners
 SELECT Cook_idCook
 INTO winner_cook_id
 FROM FinalWinners
 ORDER BY RAND()
 LIMIT 1;

 -- Update the Episode table with the cook_id of the winner for the specified episode
 UPDATE Episode
 SET winner_id = winner_cook_id
 WHERE idEpisode = inputEpisodeID;

 -- Clean up temporary tables
 DROP TEMPORARY TABLE IF EXISTS TempWinners;
 DROP TEMPORARY TABLE IF EXISTS TopScorers;
 DROP TEMPORARY TABLE IF EXISTS FinalWinners;

END //
```

DELIMITER ;

Η παραπάνω διαδικασία υπολογίζει το συνολικό score για κάθε διαγωνιζόμενο από όλους τους κριτές και ύστερα επιλέγει τους διαγωνιζόμενους με την μεγαλύτερη βαθμολογία. Αν βρει ισοπαλία τότε επιλέγει μεταξύ αυτών τους πιο εξειδικευμένους μάγειρες εξ αυτών. Αν βρει και εκεί ισοπαλία επιλέγει τυχαία έναν από τους προηγούμενους μάγειρες.

Φυσικά ο υπολογισμός θα γίνεται κάθε φορά που αλλάζουν τα δεδομένα μας για να διατηρηθεί η ακεραιότητα του νικητή το οποίο επιτυγχάνουν:

```
DELIMITER //
```

- ```
CREATE TRIGGER AfterInsertJudgeParticipantScores
AFTER INSERT ON Judge_Participant_Scores
FOR EACH ROW
BEGIN
    -- Call the procedure to calculate the winner for the episode
    CALL CalculateEpisodeWinner(NEW.Episode_idEpisode);
END //
```

```
DELIMITER ;
```

```
DELIMITER //
```

- ```
CREATE TRIGGER AfterUpdateJudgeParticipantScores
AFTER UPDATE ON Judge_Participant_Scores
FOR EACH ROW
BEGIN
 -- Call the procedure to calculate the winner for the episode
 CALL CalculateEpisodeWinner(NEW.Episode_idEpisode);
END //
```

```
DELIMITER ;
```

Οι οποίες πραγματοποιούνται κάθε φορά αφού προσθέσουμε καινούργια βαθμολογία και κάθε φορά που τροποποιήσουμε κάποια από αυτές.

Τέλος, θέλουμε εκτός του διαγωνισμού και κάποιους χρήστες της βάσεις που θα είναι είτε διαχειριστές είτε οι μάγειρες.

- ```
CREATE TABLE `Cooking_Competition`.`Admin` (
  `idAdmin` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `username` VARCHAR(55) NOT NULL,
  `password` VARCHAR(12) NOT NULL,
  `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (idAdmin))
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

- ```

CREATE TABLE Cooking_Competition.`User` (
 `idUser` INT UNSIGNED NOT NULL AUTO_INCREMENT,
 `username` VARCHAR(55) NOT NULL,
 `password` VARCHAR(12) NOT NULL,
 `Cook_idCook` INT UNSIGNED NOT NULL,
 `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
 PRIMARY KEY (`idUser`),
 INDEX `k_Cook_User_Cook1_idx` (`Cook_idCook` ASC),
 CONSTRAINT `k_Cook_User_Cuisine1`
 FOREIGN KEY (`Cook_idCook`)
 REFERENCES `Cooking_Competition`.`Cook` (`idCook`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE)
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

Οι παραπάνω πίνακες παίρνουν Primary Key τα “idUser” και “idAdmin” αντίστοιχα που είναι μοναδικά για κάθε χρήστη, ένα username και ένα password και την τελευταία τροποποίηση. Η μόνη διαφορά τους είναι ότι ο απλός User είναι ταυτόχρονα και μάγειρας μέσα στον διαγωνισμό, οπότε για να τους συσχετίσουμε θα ορίσουμε το Foreign Key “**Cook\_idCook**” το οποίο με μοναδικό τρόπο ενώνει κάθε χρήστη με έναν μάγειρα. Φυσικά κατά την δημιουργία καινούργιου χρήστη θα πρέπει να ψάχνουμε το όνομα του στους μάγειρες και να εισάγουμε το id του σαν μάγειρα.

Τώρα θα δώσουμε δικαιώματα στους χρήστες μας.

```

DELIMITER //

CREATE PROCEDURE AddAdminUser(
 IN username VARCHAR(45),
 IN user_password VARCHAR(45)
)
BEGIN
 -- Create the admin user
 SET @query = CONCAT('CREATE USER ', username, '@''localhost'' IDENTIFIED BY ', user_password, '');
 PREPARE stmt FROM @query;
 EXECUTE stmt;
 DEALLOCATE PREPARE stmt;

 -- Grant all privileges to the admin user
 SET @query = CONCAT('GRANT ALL PRIVILEGES ON *.* TO ', username, '@''localhost'' WITH GRANT OPTION;');
 PREPARE stmt FROM @query;
 EXECUTE stmt;
 DEALLOCATE PREPARE stmt;

 -- Reload the privileges
 FLUSH PRIVILEGES;

 -- Insert into Users table with the role 'administrator'
 INSERT INTO `Admin` (username, password, role) VALUES (username, user_password, 'administrator');
END //

DELIMITER ;

```

Για τον Admin αρκεί να γίνει η εισαγωγή του μέσω τις παραπάνω διαδικασίας και θα αποκτήσει όλα τα δικαιώματα συμπεριλαμβανομένου της τροποποίησης και εισαγωγής δεδομένων σε πίνακες αλλά και του backup και restore της βάσης που αυτό γίνεται μέσα από το shell του λογισμικού με τις εντολές

```
mysqldump -u (admin_username) -p (admin_password) Cooking_Competition > /path/to/backup.sql
```

```
mysql -u (admin_username) -p (admin_password) Cooking_Competition < /path/to/backup.sql
```

Για τους μάγειρες, επειδή έχουν πιο περιορισμένα δικαιώματα που αφορούν το idcook τους θα χρειαστώ αρχικά

- ```
-- -----  
-- View to show only recipes assigned to a cook  
CREATE VIEW CookAssignedRecipes AS  
SELECT  
    r.idRecipe,  
    r.title,  
    r.description,  
    r.difficulty,  
    r.prep_time,  
    r.cook_time,  
    r.portions,  
    r.total_calories,  
    r.total_fat,  
    r.total_protein,  
    r.total_carbohydrate,  
    r.Cuisine_id,  
    r.Type_Meal,  
    r.syntagi,  
    r.last_update,  
    r.Image,  
    ep.Cook_idCook  
FROM  
    Recipe r  
JOIN  
    Episode_has_Participants ep ON r.idRecipe = ep.Recipe_idRecipe;
```

-- View to show personal details of a cook

● CREATE VIEW CookPersonalDetails AS

SELECT

idCook,
first_name,
last_name,
phone_number,
birth_date,
age,
Years_experience,
Status,
last_update,
Image


FROM

Cook;

-- Procedure to update recipes assigned to a cook

DELIMITER //

DELIMITER //

●  CREATE PROCEDURE UpdateCookDetails(
IN username VARCHAR(45),
IN new_first_name VARCHAR(45),
IN new_last_name VARCHAR(45),
IN new_phone_number VARCHAR(15),
IN new_birth_date DATE,
IN new_age INT,
IN new_Years_experience INT,
IN new_Status ENUM('C cook', 'B cook', 'A cook', 'assistant head Chef', 'Chef'),
IN new_Image INT
)

 BEGIN

DECLARE cook_id INT;

-- Get the cook_id from the Users and Cook tables

SELECT c.idCook
INTO cook_id
FROM Users u
INNER JOIN Cook c ON u.idUser = c.User_idUser
WHERE u.username = username;

-- Update the cook's details

UPDATE Cook

SET

first_name = new_first_name,
last_name = new_last_name,
phone_number = new_phone_number,
birth_date = new_birth_date,
age = new_age,
Years_experience = new_Years_experience,
Status = new_Status,
Image = new_Image,
last_update = CURRENT_TIMESTAMP

WHERE idCook = cook_id;

END //

DELIMITER ;

```
-- Procedure to update recipes assigned to a cook  
DELIMITER //
```

```
CREATE PROCEDURE UpdateAssignedRecipe(  
    IN username VARCHAR(45),  
    IN recipe_id INT,  
    IN new_title VARCHAR(200),  
    IN new_description TEXT,  
    IN new_difficulty TINYINT,  
    IN new_prep_time TINYINT,  
    IN new_cook_time TINYINT,  
    IN new_portions TINYINT,  
    IN new_total_calories DECIMAL(10,2),  
    IN new_total_fat DECIMAL(10,2),  
    IN new_total_protein DECIMAL(10,2),  
    IN new_total_carbohydrate DECIMAL(10,2),  
    IN new_Cuisine_id INT,  
    IN new_Type_Meal INT,  
    IN new_syntagi ENUM('cooking', 'pastry'),  
    IN new_Image INT  
)  
  
BEGIN  
    DECLARE cook_id INT;  
    DECLARE is_assigned INT;  
  
    -- Get the cook_id from the Users and Cook tables  
    SELECT c.idCook  
    INTO cook_id  
    FROM Users u  
    INNER JOIN Cook c ON u.idUser = c.User_idUser  
    WHERE u.username = username;  
  
    -- Check if the cook is assigned to the recipe  
    SELECT COUNT(*)  
    INTO is_assigned  
    FROM Episode_has_Participants  
    WHERE Cook_idCook = cook_id AND Recipe_idRecipe = recipe_id;  
  
    IF is_assigned = 0 THEN  
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cook not assigned to this recipe';  
    ELSE  
        UPDATE Recipe  
        SET  
            title = new_title,  
            description = new_description,  
            difficulty = new_difficulty,  
            prep_time = new_prep_time,  
            cook_time = new_cook_time,  
            portions = new_portions,  
            total_calories = new_total_calories,  
            total_fat = new_total_fat,  
            total_protein = new_total_protein,  
            total_carbohydrate = new_total_carbohydrate,  
            Cuisine_id = new_Cuisine_id,  
            Type_Meal = new_Type_Meal,  
            syntagi = new_syntagi,  
            Image = new_Image,  
            last_update = CURRENT_TIMESTAMP  
        WHERE idRecipe = recipe_id;  
    END IF;  
END //
```

```
DELIMITER ;
```

Οι παραπάνω διαδικασίες είναι οι υπο-διαδικασίες όπου θα δώσουμε δικαιώματα στους μάγειρες-χρήστες να τρέχουν για να επεξεργάζονται τα προσωπικά τους δεδομένα και τις συνταγές που μπορούν να εκτελέσουν. Αρχικά δημιουργούμε ένα view (έναν υπο-πίνακα του κανονικού στον οποίο επιλέγουμε να βλέπει ο χρήστης μόνο ότι περιέχει το id του) τον οποίο μετά λόγω κατασκευής του και της ιδιότητας του σαν view θα πρέπει να μπορεί να τροποποιεί μέσω τις διαδικασίας η οποία συγκρίνει τι βλέπει ο χρήστης και τι αλλάζει πάνω σε αυτό με το τι υπάρχει πραγματικά από κάτω.

Τέλος θα έχουμε και την διαδικασία προσθήκης καινούργιου μάγειρα-χρήστη κατά την οποία του δίνουμε την δυνατότητα να βλέπει και να επεξεργάζεται τα προσωπικά του στοιχεία μόνο και τις συνταγές του μόνο (όπου συνταγές του θεωρούνται όλες οι συνταγές που ανήκουν σε κάθε κουζίνα στην οποία έχει εξειδίκευση) και την δυνατότητα να προσθέτει καινούργιες συνταγές.

```
DELIMITER //
```

```
CREATE PROCEDURE AddCookUser(  
    IN username VARCHAR(45),  
    IN user_password VARCHAR(45),  
    IN cook_id INT  
)  
BEGIN  
    -- Create the user  
    SET @query = CONCAT('CREATE USER \'', username, '\'@\'localhost\' IDENTIFIED BY \'', user_password, '\';');  
    PREPARE stmt FROM @query;  
    EXECUTE stmt;  
    DEALLOCATE PREPARE stmt;  
  
    -- Grant privileges to the cook user for the views  
    SET @query = CONCAT('GRANT SELECT, UPDATE ON Cooking_Competition.CookAssignedRecipes TO \'', username, '\'@\'localhost\';');  
    PREPARE stmt FROM @query;  
    EXECUTE stmt;  
    DEALLOCATE PREPARE stmt;  
  
    SET @query = CONCAT('GRANT SELECT, UPDATE ON Cooking_Competition.CookPersonalDetails TO \'', username, '\'@\'localhost\';');  
    PREPARE stmt FROM @query;  
    EXECUTE stmt;  
    DEALLOCATE PREPARE stmt;  
  
    -- Grant execution privileges for the stored procedures  
    SET @query = CONCAT('GRANT EXECUTE ON PROCEDURE Cooking_Competition.UpdateAssignedRecipe TO \'', username, '\'@\'localhost\';');  
    PREPARE stmt FROM @query;  
    EXECUTE stmt;  
    DEALLOCATE PREPARE stmt;  
  
    SET @query = CONCAT('GRANT EXECUTE ON PROCEDURE Cooking_Competition.UpdateCookDetails TO \'', username, '\'@\'localhost\';');  
    PREPARE stmt FROM @query;  
    EXECUTE stmt;  
    DEALLOCATE PREPARE stmt;
```



```

-- Grant INSERT privileges on the specified tables
SET @query = CONCAT('GRANT INSERT ON Cooking_Competition.Recipe TO \'', username, '\'@\'localhost\';');
PREPARE stmt FROM @query;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;

SET @query = CONCAT('GRANT INSERT ON Cooking_Competition.Recipe_has_Concept TO \'', username, '\'@\'localhost\';');
PREPARE stmt FROM @query;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;

SET @query = CONCAT('GRANT INSERT ON Cooking_Competition.Recipe_has_Label TO \'', username, '\'@\'localhost\';');
PREPARE stmt FROM @query;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;

SET @query = CONCAT('GRANT INSERT ON Cooking_Competition.Recipe_has_Equipment TO \'', username, '\'@\'localhost\';');
PREPARE stmt FROM @query;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;

SET @query = CONCAT('GRANT INSERT ON Cooking_Competition.Tips TO \'', username, '\'@\'localhost\';');
PREPARE stmt FROM @query;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;

SET @query = CONCAT('GRANT INSERT ON Cooking_Competition.Steps TO \'', username, '\'@\'localhost\';');
PREPARE stmt FROM @query;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;

-- Reload the privileges
FLUSH PRIVILEGES;
INSERT INTO User (username , password, Cook_idCook ) VALUES (username , user_password, cook_id );

END //

DELIMITER ;

```

Για να μπορεί να προσθέτει συνταγές πρέπει ο χρήστης να έχει πρόσβαση εισαγωγής στους ενδιάμεσους πίνακες της συνταγής με τους άλλους πίνακες αλλά και στον πίνακα συνταγή, βήματα και συμβουλές.

Γ)

Για την δημιουργία της βάσης μας χρησιμοποιήσαμε το **MySQL Community** [εδώ](#) και για την επιλογή των δεδομένων μας για τα επεισόδιά και την εισαγωγή των βαθμολογιών χρησιμοποιήσαμε την `python` και την βιβλιοθήκη της **mysql-connector-python 8.4.0**. Για την διαχείριση και την ανάπτυξη της βάσης χρησιμοποιήσαμε και το **MySQL Workbench** [εδώ](#).

Δ)

Αρχικά θα κατεβάσουμε το installer της MySQL Community και θα το τρέξουμε και θα κατεβάσουμε τα MySQL Server 8.0.37

MySQL Shell 8.0.37

MySQL Router 8.0.37

Και ύστερα κατεβάζουμε και τρέχουμε το installer του MySQL Workbench 8.0.36.

Χρειαζόμαστε επίσης στον υπολογιστή μας εγκατεστημένη python (δουλεύουν και πιο παλιές versions) και ανοίγουμε το **cmd** και τρέχουμε την εντολή **pip install mysql-connector-python** για να κατεβάσουμε την βιβλιοθήκη.

Έχοντας πλέον όλα μας τα προγράμματα, είμαστε έτοιμοι να αρχίσουμε να τρέχουμε την βάση μας.

Ανοίγουμε ένα **cmd** και μπαίνουμε στο path που έχουμε κατεβάσει την sql π.χ.

```
cd C:\Program Files\MySQL\MySQL Server 8.0\bin
```

και μπαίνουμε στην βάση με

```
mysql -u root -p
```

βάζουμε τον κωδικό μας αν έχουμε

και τρέχουμε την εντολή για να δημιουργήσουμε την βάση μας στο shell της sql

```
source path to scheme\scheme.sql
```

όπου path to scheme είναι το αντίστοιχο path για το που είναι αποθηκευμένο το αρχείο π.χ.

```
source C:\scheme.sql
```

Τώρα για να κάνουμε populate θα τρέξουμε στο shell της sql

```
source path to Insert data\Insert data.sql
```

π.χ. **source C:\Insert data.sql**

Υστερα θέλουμε να ανοίξουμε ένα καινούργιο cmd για να τρέξουμε τα python αρχεία μας.

Αρχικά πάμε στο path όπου τα έχουμε αποθηκεύσει π.χ.

```
cd C:\
```

και τρέχουμε για τις σεζόν

```
python generateSeasons.py
```

και για τις βαθμολογίες

```
python insert_scores.py
```

Τέλος πίσω στο shell της sql τρέχουμε για τα queries (απαντήσεις στα ερωτήματα)

```
source path to queries \ queries.sql
```

π.χ. **source C:\ queries.sql**

Μετά από τα παραπάνω είμαστε έτοιμοι να τρέξουμε τα ερωτήματα μας αλλά και να τροποποιήσουμε τα δεδομένα μας, να εισάγουμε καινούργια ή να δούμε τα ήδη υπάρχον.