

Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

***Βάσεις Δεδομένων***

*Αναφορά Εξαμηνιαίας εργασίας*

Δέσποινα – Χριστίνα Μαρκάτου (03121433)

Γιώργος Παρασκευάς Παπανδρικόπουλος (03121102)

Χαράλαμπος Σπυρόπουλος (03121122)

Περιεχόμενα:

1. Διαγράμματα Βάσης Δεδομένων (ER και Σχεσιακό διάγραμμα)
2. Ευρετήρια (Indexes)
3. Υλοποίηση της εφαρμογής (DDL και DML script)
4. Οδηγίες εγκατάστασης εφαρμογής
5. Ανάλυση λογικής queries
6. Σύνδεσμος github repository

1. Διαγράμματα Βάσης Δεδομένων

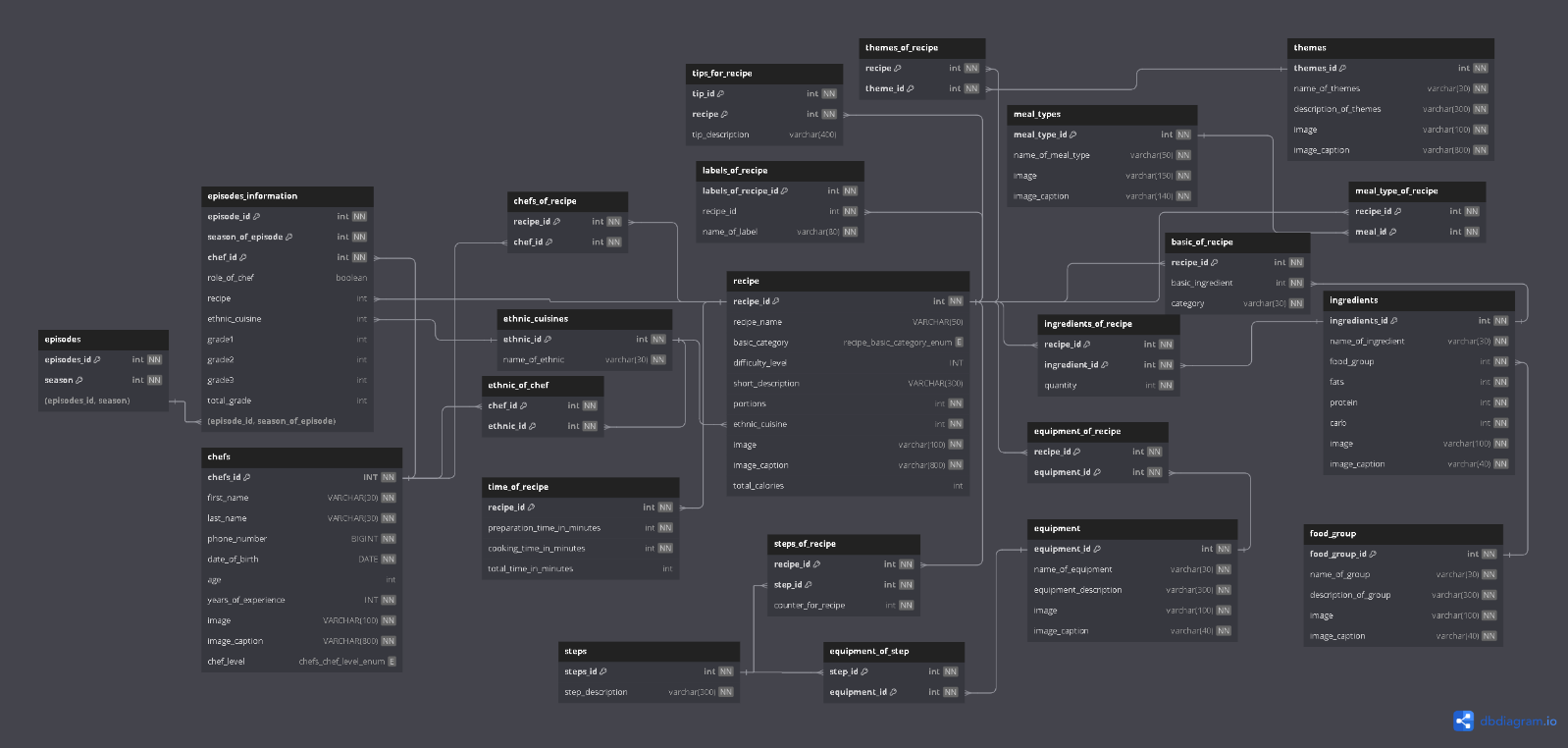
1.1 Διάγραμμα Οντοτήτων-Συσχετίσεων (Entity-Relationship Diagram)

Εικόνα που περιέχει κείμενο, διάγραμμα, Σχέδιο, τεχνικό σχέδιο

Περιγραφή που δημιουργήθηκε αυτόματαΠαρακάτω φαίνεται το ER διάγραμμα της βάσης δεδομένων που υλοποιήσαμε, σύμφωνα με τις προδιαγραφές που δόθηκαν:

Το διάγραμμα δημιουργήθηκε χρησιμοποιώντας το εργαλείο draw-io και εμφανίζει τις οντότητες (entities) της βάσης μας, τα χαρακτηριστικά τους (attributes), καθώς και τις σχέσεις (relationships) που τις συνδέουν (τα attributes που χαρακτηρίζουν με μοναδικό τρόπο τα entities συμβολίζονται με έντονο χρώμα και είναι υπογραμμισμένα).

1.2 Σχεσιακό διάγραμμα (Relational Diagram)

Παρακάτω φαίνεται το αντίστοιχο σχεσιακό διάγραμμα της βάσης δεδομένων:

Το relational model δημιουργήθηκε στην εφαρμογή [bddiagram.iο](https://dbdiagram.io/d/6651f6a8f84ecd1d222042b9) εισάγοντας το ddl script, που δημιουργήσαμε στο MySQL Workbench. Στο διάγραμμα εμφανίζονται πίνακες που αντιστοιχούν στα entities του ERD, καθώς και επιπλέον πίνακες που στο ERD απεικονίζονταν ως σχέσεις. Επιπλέον φαίνονται οι τύποι δεδομένων των μεταβλητών και αποτυπώνεται μία πιο ξεκάθαρη εικόνα ως προς των τρόπο που συνδέονται οι πίνακες με την χρήση primary και foreign keys.

2. Ευρετήρια

Η δημιουργία των indexes έχει ως στόχο να επιταχύνει την αναζήτηση και την ανάκτηση δεδομένων, αφού δεν θα είναι πλέον απαραίτητη η πλήρης σάρωση του πίνακα. Γενικά επιλέγουμε να δημιουργήσουμε indexes για τις στήλες που χρησιμοποιούνται συχνά στις εντολές ORDER BY και GROUP BY, καθώς και σε ερωτήματα που συνδυάζουν δεδομένα από πολλούς πίνακες, όπως στις εντολές JOIN. Επίσης, επιλέγουμε στήλες που μπορούν να πάρουν πολλές διαφορετικές τιμές και όχι για παράδειγμα στήλες που αντιστοιχούν σε boolean τιμές. Πρέπει όμως να λάβουμε υπόψιν ότι παρόλο που τα indexes επιταχύνουν την ανάκτηση δεδομένων, μπορούν να επιβραδύνουν τις λειτουργίες INSERT, UPDATE και DELETE, διότι το index πρέπει να ενημερώνεται κάθε φορά που αλλάζουν τα δεδομένα. Επίσης για πολύ μικρούς πίνακες, το κόστος της δημιουργίας και συντήρησης του index μπορεί να μην αντισταθμίζει το όφελος.

Έτσι, σύμφωνα με τις παραπάνω παρατηρήσεις, καθώς και τις ερωτήσεις που καλούμαστε να κάνουμε στην βάση, δημιουργήσαμε τα παρακάτω ευρετήρια:

• chefs\_id του table chefs, για γρήγορη εύρεση των chefs

• recipe\_id του table recipe, για γρήγορη εύρεση των recipes

• ingredients\_id του table ingredients, για γρήγορη εύρεση των ingredients

• recipe\_name του table recipe, διότι η αναζήτηση των συνταγών γίνεται μέσω του ονόματός τους

• (first\_name, last\_name) του table chefs, διότι η αναζήτηση των μαγείρων γίνεται μέσω του ονόματός τους

• chef\_id του table chefs\_of\_recipe, για γρήγορη εύρεση των συνταγών που μπορεί να εκτελέσει κάποιος μάγειρας

• recipe\_id του table ingredients\_of\_recipe, για γρήγορη εύρεση των υλικών που απαιτούνται για την εκτέλεση μίας συνταγής

• chef\_id του table ethnics\_of\_chef, για γρήγορη εύρεση των εθνικών κουζινών που αντιστοιχούν σε κάποιον μάγειρα

• chef\_id του table episodes\_information, διότι σε πολλά ερωτήματα πρέπει να εντοπίσουμε τους μάγειρες που συμμετέχουν σε κάποιο επεισόδιο

• ethnic\_cuisine του table episodes\_information, διότι σε πολλά ερωτήματα πρέπει να εντοπίσουμε τις εθνικές κουζίνες που εμφανίζονται σε κάποιο επεισόδιο

• recipe του table episodes\_information, διότι σε πολλά ερωτήματα πρέπει να εντοπίσουμε τις συνταγές που εκτελούνται σε κάποιο επεισόδιο

• season\_of\_episode του table episodes\_information, διότι σε πολλά ερωτήματα πρέπει να χωρίσουμε τα αποτελέσματα με βάση τη σεζόν στην οποία ανήκουν

• recipe\_id του table equipment\_of\_recipe, για γρήγορη εύρεση του εξοπλισμού που απαιτείται για την εκτέλεση κάποιας συνταγής

• recipe\_id του table meal\_type\_of\_recipe, για γρήγορη εύρεση των τύπων γεύματος που αντιστοιχούν σε κάποια συνταγή

• recipe\_id του table steps\_of\_recipe, για γρήγορη εύρεση των βημάτων που πρέπει να ακολουθηθούν για την εκτέλεση μίας συνταγής

• themes\_id του table themes, για γρήγορη εύρεση των θεματικών ενοτήτων

• recipe του table themes\_of\_recipe, για γρήγορη εύρεση των θεματικών ενοτήτων στις οποίες εντάσσεται μία συνταγή

• (recipe\_id, name\_of\_label) του table labels\_of\_recipe, για γρήγορη εύρεση των ετικετών που αντιστοιχούν σε μία συνταγή, καθώς και το όνομά τους

Στα ευρετήρια συμπεριλάβαμε και κάποια που αναφέρονται στο primary key ενός table για λόγους πληρότητας, όμως για τα primary keys δημιουργείται έτσι και αλλιώς ευρετήριο κατά τον ορισμό τους. Υπάρχει πιθανότητα με την πάροδο του χρόνου να παρατηρηθούν διαφορετικές απαιτήσεις όσον αφορά στην συχνότητα εμφάνισης των διάφορων ερωτήματων, οπότε και να χρειαστεί να αλλάξουμε τα indexes με βάση αυτές.

3. Υλοποίηση εφαρμογής

Για τον σχεδιασμό και την υλοποίηση της βάσης δεδομένων χρησιμοποιήθηκε το σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων mySQL.

3.1 DDL script

Ακολουθεί ο κώδικας σε sql του DDL script για την δημιουργία των πινάκων της βάσης μας, μαζί με τους περιορισμούς ακεραιότητας στα διάφορα χαρακτηριστικά των πινάκων. Για την δημιουργία της βάσης έχουμε χρησιμοποιήσει και διάφορα triggers και procedures, για να ελέγχουμε την σωστή εισαγωγή δεδομένων στην βάση μας, καθώς και την σωστή τελική κατάσταση της βάσης. Αυτά τα μέρη του κώδικα περιέχονται στο git repo μας στους ακόλουθους συνδέσμους:

[https://github.com/Madeperf/DataBaseProject/blob/main/DDLscript.sql](https://github.com/Madeperf/DataBaseProject/blob/main/DDLscript.sql%20)

[https://github.com/Madeperf/DataBaseProject/blob/main/Procedures.sql](https://github.com/Madeperf/DataBaseProject/blob/main/Procedures.sql%20)

*DROP SCHEMA if exists `masterchef`;*

*CREATE SCHEMA `masterchef`;*

*use masterchef;*

*-- Απενεργοποίηση των περιορισμών ξένων κλειδιών*

*SET foreign\_key\_checks = 0;*

*-- Διαγραφή όλων των πινάκων*

*DROP TABLE IF EXISTS ethnic\_cuisines;*

*DROP TABLE IF EXISTS recipe;*

*DROP TABLE IF EXISTS tips\_for\_recipe;*

*DROP TABLE IF EXISTS meal\_types;*

*DROP TABLE IF EXISTS meal\_type\_of\_recipe;*

*DROP TABLE IF EXISTS steps;*

*DROP TABLE IF EXISTS steps\_of\_recipe;*

*DROP TABLE IF EXISTS equipment;*

*DROP TABLE IF EXISTS equipment\_of\_recipe;*

*DROP TABLE IF EXISTS equipment\_of\_step;*

*DROP TABLE IF EXISTS food\_group;*

*DROP TABLE IF EXISTS ingredients;*

*DROP TABLE IF EXISTS ingredients\_of\_recipe;*

*DROP TABLE IF EXISTS time\_of\_recipe;*

*DROP TABLE IF EXISTS themes;*

*DROP TABLE IF EXISTS themes\_of\_recipe;*

*DROP TABLE IF EXISTS chefs;*

*DROP TABLE IF EXISTS chefs\_of\_recipe;*

*DROP TABLE IF EXISTS ethnic\_of\_chef;*

*DROP TABLE IF EXISTS labels\_of\_recipe;*

*DROP TABLE IF EXISTS basic\_of\_recipe;*

*DROP TABLE IF EXISTS episodes;*

*DROP TABLE IF EXISTS episodes\_information;*

*SET foreign\_key\_checks = 1;*

*create table ethnic\_cuisines(*

*ethnic\_id int UNSIGNED NOT NULL AUTO\_INCREMENT primary key ,*

*name\_of\_ethnic varchar(30) UNIQUE NOT NULL*

*);*

*create table recipe(*

*recipe\_id int UNSIGNED NOT NULL AUTO\_INCREMENT PRIMARY KEY ,*

*recipe\_name VARCHAR(50) ,*

*basic\_category ENUM("savory","pastry"),*

*difficulty\_level INT UNSIGNED check(difficulty\_level BETWEEN 1 AND 5),*

*short\_description VARCHAR(300),*

*portions int NOT NULL ,*

*ethnic\_cuisine int unsigned NOT NULL,*

*image varchar(100) not null,*

*image\_caption varchar(800) not null,*

*total\_calories int,*

*CONSTRAINT FK\_recipe\_ethnic foreign key (ethnic\_cuisine) references ethnic\_cuisines(ethnic\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE*

*);*

*create table tips\_for\_recipe(*

*tip\_id int NOT NULL,*

*recipe int Unsigned NOT NULL,*

*tip\_description varchar(400),*

*primary key( tip\_id,recipe) ,*

*CONSTRAINT tips\_for\_recipe\_recipe\_id foreign key(recipe) references recipe(recipe\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE*

*);*

*create table meal\_types (*

*meal\_type\_id int primary key NOT NULL,*

*name\_of\_meal\_type varchar(50) NOT NULL,*

*image varchar(150) not null ,*

*image\_caption varchar(140) not null*

*);*

*create table meal\_type\_of\_recipe(*

*recipe\_id int unsigned NOT NULL,*

*meal\_id int NOT NULL,*

*primary key(recipe\_id,meal\_id),*

*CONSTRAINT meal\_type\_of\_recipe\_recipe\_id foreign key(recipe\_id) references recipe(recipe\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE,*

*CONSTRAINT meal\_type\_of\_recipe\_meal\_id foreign key(meal\_id) references meal\_types(meal\_type\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE*

*);*

*create table steps (*

*steps\_id int UNSIGNED NOT NULL AUTO\_INCREMENT primary key ,*

*step\_description varchar(300) NOT NULL*

*);*

*create table steps\_of\_recipe(*

*recipe\_id int unsigned NOT NULL ,*

*step\_id int unsigned NOT NULL,*

*counter\_for\_recipe int NOT NULL,*

*primary key(recipe\_id,step\_id),*

*CONSTRAINT steps\_of\_recipe\_recipe\_id foreign key(recipe\_id) references recipe(recipe\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE,*

*CONSTRAINT steps\_of\_recipe\_step\_id foreign key(step\_id) references steps(steps\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE*

*);*

*create table equipment (*

*equipment\_id int UNSIGNED NOT NULL AUTO\_INCREMENT primary key ,*

*name\_of\_equipment varchar(30) NOT NULL,*

*equipment\_description varchar(300) NOT NULL,*

*image varchar(100) not null ,*

*image\_caption varchar(40) not null*

*);*

*create table equipment\_of\_recipe(*

*recipe\_id int unsigned NOT NULL,*

*equipment\_id int unsigned NOT NULL,*

*primary key(recipe\_id,equipment\_id),*

*CONSTRAINT equipment\_of\_recipe\_recipe\_id foreign key(recipe\_id) references recipe(recipe\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE,*

*CONSTRAINT equipment\_of\_recipe\_equipment\_id foreign key(equipment\_id) references equipment(equipment\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE);*

*create table equipment\_of\_step(*

*step\_id int unsigned NOT NULL,*

*equipment\_id int unsigned NOT NULL,*

*primary key(step\_id,equipment\_id),*

*CONSTRAINT equipment\_of\_step\_step\_id foreign key(step\_id) references steps(steps\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE,*

*CONSTRAINT equipment\_of\_step\_equipment\_id foreign key(equipment\_id) references equipment(equipment\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE*

*);*

*create table food\_group (*

*food\_group\_id int primary key NOT NULL,*

*name\_of\_group varchar(30) NOT NULL,*

*description\_of\_group varchar(300) NOT NULL,*

*image varchar(100) not null,*

*image\_caption varchar(40) not null*

*);*

*create table ingredients (*

*ingredients\_id int UNSIGNED NOT NULL AUTO\_INCREMENT primary key ,*

*name\_of\_ingredient varchar(30) NOT NULL,*

*food\_group int NOT NULL,*

*fats int NOT NULL,*

*protein int NOT NULL,*

*carb int NOT NULL,*

*image varchar(100) not null,*

*image\_caption varchar(40) not null,*

*CONSTRAINT ingredients\_food\_group foreign key (food\_group) references food\_group(food\_group\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE*

*);*

*create table ingredients\_of\_recipe(*

*recipe\_id int unsigned NOT NULL,*

*ingredient\_id int unsigned NOT NULL,*

*quantity int NOT NULL,*

*primary key(recipe\_id,ingredient\_id),*

*CONSTRAINT ingredients\_of\_recipe\_recipe\_id foreign key(recipe\_id) references recipe(recipe\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE,*

*CONSTRAINT ingredients\_of\_recipe\_ingredient\_id foreign key(ingredient\_id) references ingredients(ingredients\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE*

*);*

*create table time\_of\_recipe (*

*recipe\_id int unsigned NOT NULL,*

*preparation\_time\_in\_minutes int NOT NULL,*

*cooking\_time\_in\_minutes int NOT NULL,*

*total\_time\_in\_minutes int ,*

*primary key (recipe\_id),*

*CONSTRAINT time\_of\_recipe\_recipe\_id foreign key(recipe\_id) references recipe(recipe\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE*

*);*

*create table themes(*

*themes\_id int UNSIGNED NOT NULL AUTO\_INCREMENT primary key ,*

*name\_of\_themes varchar(30) NOT NULL,*

*description\_of\_themes varchar(300) NOT NULL,*

*image varchar(100) not null,*

*image\_caption varchar(800) not null*

*);*

*create table themes\_of\_recipe(*

*recipe int unsigned NOT NULL,*

*theme\_id int unsigned NOT NULL,*

*primary key(recipe,theme\_id),*

*CONSTRAINT themes\_of\_recipe\_recipe\_id foreign key(recipe) references recipe(recipe\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE,*

*CONSTRAINT themes\_of\_recipe\_theme\_id foreign key(theme\_id) references themes(themes\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE*

*);*

*CREATE TABLE chefs (*

*chefs\_id INT UNSIGNED NOT NULL AUTO\_INCREMENT PRIMARY KEY,*

*first\_name VARCHAR(30) NOT NULL,*

*last\_name VARCHAR(30) NOT NULL,*

*phone\_number BIGINT UNIQUE NOT NULL CHECK (phone\_number BETWEEN 6900000000 AND 6999999999),*

*date\_of\_birth DATE NOT NULL,*

*age int unsigned,*

*years\_of\_experience INT NOT NULL,*

*image VARCHAR(100) NOT NULL ,*

*image\_caption VARCHAR(800) NOT NULL,*

*chef\_level ENUM('G CHEF ', 'B CHEF', 'A CHEF', 'ASSISTANT CHEF', 'MASTER CHEF')*

*);*

*create table chefs\_of\_recipe(*

*recipe\_id int unsigned NOT NULL ,*

*chef\_id int unsigned NOT NULL,*

*primary key(recipe\_id,chef\_id),*

*CONSTRAINT chefs\_of\_recipe\_recipe\_id foreign key(recipe\_id) references recipe(recipe\_id),*

*CONSTRAINT chefs\_of\_recipe\_chef\_id foreign key(chef\_id) references chefs(chefs\_id)*

*);*

*create table ethnic\_of\_chef(*

*chef\_id int unsigned NOT NULL,*

*ethnic\_id int unsigned NOT NULL,*

*primary key(ethnic\_id,chef\_id),*

*CONSTRAINT ethnic\_of\_chef\_ethnic\_id foreign key(ethnic\_id) references ethnic\_cuisines(ethnic\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE,*

*CONSTRAINT ethnic\_of\_chef\_chef\_id foreign key(chef\_id) references chefs(chefs\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE*

*);*

*create table labels\_of\_recipe(*

*labels\_of\_recipe\_id int UNSIGNED NOT NULL AUTO\_INCREMENT ,*

*recipe\_id int unsigned NOT NULL,*

*name\_of\_label varchar(80) NOT NULL ,*

*primary key(labels\_of\_recipe\_id) ,*

*CONSTRAINT labels\_of\_recipe\_recipe\_id foreign key(recipe\_id) references recipe(recipe\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE,*

*UNIQUE KEY (recipe\_id, name\_of\_label)*

*) ;*

*create table basic\_of\_recipe(*

*recipe\_id int unsigned NOT NULL,*

*basic\_ingredient int unsigned NOT NULL,*

*category varchar(30) NOT NULL,*

*primary key(recipe\_id) ,*

*CONSTRAINT basic\_of\_recipe\_recipe\_id foreign key(recipe\_id) references recipe(recipe\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE,*

*CONSTRAINT basic\_of\_recipe\_basic\_ingredient foreign key(basic\_ingredient) references ingredients(ingredients\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE*

*);*

*create table episodes(*

*episodes\_id int UNSIGNED NOT NULL AUTO\_INCREMENT,*

*season int UNSIGNED NOT NULL,*

*primary key(episodes\_id,season)*

*);*

*create table episodes\_information(*

*episode\_id int unsigned NOT NULL,*

*season\_of\_episode int unsigned not null,*

*chef\_id int unsigned NOT NULL,*

*role\_of\_chef boolean, -- role of chef = 1 μαγειρας / role of chef = 0 κριτης*

*recipe int unsigned,*

*ethnic\_cuisine int unsigned,*

*grade1 int check(grade1 BETWEEN 1 AND 5),*

*grade2 int check(grade2 BETWEEN 1 AND 5),*

*grade3 int check(grade3 BETWEEN 1 AND 5),*

*total\_grade int UNSIGNED,*

*primary key (episode\_id,season\_of\_episode,chef\_id),*

*CONSTRAINT episodes\_information\_episode\_id foreign key (episode\_id,season\_of\_episode) references episodes(episodes\_id,season)*

*ON DELETE RESTRICT ON UPDATE CASCADE,*

*CONSTRAINT episodes\_information\_chef\_id foreign key (chef\_id) references chefs(chefs\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE,*

*CONSTRAINT episodes\_information\_recipe foreign key (recipe) references recipe(recipe\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE,*

*CONSTRAINT episodes\_information\_ethnic\_cuisine foreign key (ethnic\_cuisine) references ethnic\_cuisines(ethnic\_id)*

*ON DELETE RESTRICT ON UPDATE CASCADE*

*);*

Στον κώδικα μας, αφού δημιουργήσουμε το σχήμα με όνομα “masterchef” για την βάση μας, με την χρήση της εντολής create table δημιουργούμε τους πίνακες που θα χρησιμοποιήσουμε. Η σειρά δημιουργίας των πινάκων μας είναι τέτοια έτσι ώστε τυχών πίνακες που περιέχουν ξένα κλειδιά να δημιουργούνται αφού έχουν δημιουργηθεί οι πίνακες στους οποίους αναφέρονται.

Στα αρχεία ddl.sql και procedures.sql που βρίσκονται στο github repository, περιέχονται και τα απαραίτητα triggers που εξασφαλίζουν την ορθότητα της βάσης (π.χ. η ηλικία ενός μάγειρα να είναι μεγαλύτερη από τα χρόνια εμπειρίας του, να μην εμφανίζεται η ίδια εθνική κουζίνα σε περισσότερα από τρία διαδοχικά επεισόδια κ.τ.λ.), καθώς και τα απαραίτητα procedures (π.χ. για την ανάδειξη του νικητή του διαγωνισμού).

3.2 DML script

Με το DML script, μέσω της εντολής insert, αποθηκεύουμε δεδομένα στην βάση μας. Ενδεικτικά έχουμε δημιουργήσει 100 συνταγές, 100 μάγειρες, 50 θεματικές ενότητες (themes), 200 ετικέτες (labels), 100 συστατικά, 100 κομμάτια εξοπλισμού (equipment), 20 εθνικές κουζίνες, 50 επεισόδια και 5 σεζόν. Το DML script βρίσκεται στον ακόλουθο σύνδεσμο: <https://github.com/Madeperf/DataBaseProject/blob/main/Fake_data.sql>

Τα mock data που εισαγάγαμε στη βάση δεδομένων, κατασκευάστηκαν με χρήση του διαδικτυακού εργαλείου Mockaroo (https://www.mockaroo.com/).

4. Οδηγίες Εγκατάστασης

Για να δημιουργήσετε και να χρησιμοποιήσετε την βάση δεδομένων που έχουμε υλοποιήσει, θα πρέπει αρχικά να έχετε εγκαταστήσει την κατάλληλη έκδοση του MySQL ανάλογα με το λειτουργικό σας σύστημα (Windows, macOS, Linux) από την επίσημη ιστοσελίδα της MySQL. Κατά τη διάρκεια της εγκατάστασης, θα σας ζητηθεί να ορίσετε τον root κωδικό πρόσβασης. Μετά την εγκατάσταση, πρέπει να εκκινήσετε τον MySQL Server και να πλοηγηθείτε στο directory C:\Program Files\MySQL\MySQL Server 8.1\bin (το directory ίσως διαφέρει ανάλογα με το λειτουργικό σας σύστημα) πληκτρολογώντας στο τερματικό:

cd C:\Program Files\MySQL\MySQL Server 8.1\bin

Στην συνέχεια πρέπει να συνδεθείτε ως root χρήστης πληκτρολογώντας στο τερματικό:

mysql -u root -p

Θα σας ζητηθεί να εισαγάγετε τον root κωδικό πρόσβασης που ορίσατε κατά την εγκατάσταση. Μετά τη σύνδεση, θα πρέπει να εκτελέστε με τη σειρά τα scripts ddl.sql, dummy.sqlκαι procedures.sql τα οποία κατασκευάζουν τους πίνακες με τα ευρετήρια, τα απαραίτητα procedures και εισάγουν τα δεδομένα στη βάση. Για να το κάνετε αυτό πρέπει να εκτελέσετε στο τερματικό την εντολή:

source <path>

όπου το path θα το αντικαταστήσετε με το μονοπάτι που αντιστοιχεί σε κάθε αρχείο sql. Αυτό προυποθέτει να έχετε κατεβάσει τα αντίστοιχα αρχεία από το github repository.

4.1 Authentication

Ως root χρήστης έχετε όλα τα permissions για τροποποίηση της βάσης δεδομένων. Μέσω του ddl script έχουν δημιουργηθεί επίσης ένας χρήστης admin, αλλά και ένας χρήστης chef, που αντιστοιχεί στον chef με chef\_id=50. Για κάθε χρήστη, το σύστημα εξακριβώνει την ταυτότητά του κατά την είσοδο στην εφαρμογή (μέσω username / password). Συγκεκριμένα έχουν οριστεί:

• admin

username: admin

password: admin

• chef user

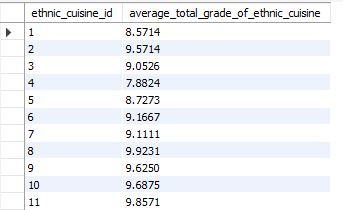
username: chef

password: chef

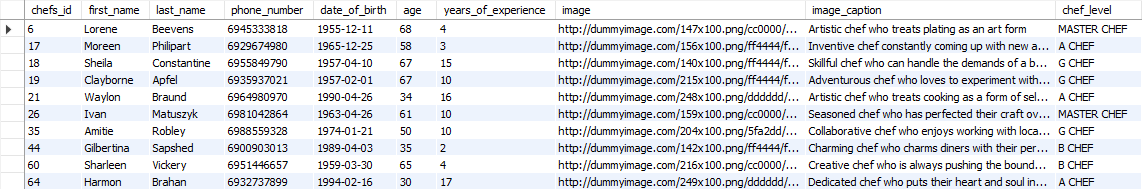
Ο admin μπορεί να καταχωρίζει και να τροποποιεί όλα τα απαιτούμενα στοιχεία. Μπορεί ακόμη να δημιουργήσει αντίγραφο ασφαλείας για όλη τη βάση (backup) και να επαναφέρει το σύστημα από αυτό (restore). O χρήστης chef έχει την δυνατότητα να επεξεργαστεί τα στοιχεία των συνταγών που του αντιστοιχούν, καθώς και να προσθέσει νέα συνταγή. Επίσης μπορεί να επεξεργαστεί τα προσωπικά του στοιχεία. Οι δύο αυτές λειτουργίες γίνονται με κλήση των procedures InsertOrUpdateRecipe και UpdateChefInfo αντίστοιχα με τις κατάλληλες παραμέτρους. Σε περίπτωση που θέλαμε να φτιάξουμε και άλλους chef users, που αντιστοιχούν στα υπόλοιπα chef\_ids της βάσης μας, θα έπρεπε να κάνουμε την ίδια διαδικασία για κάθε id (να φτιάξουμε user με κατάλληλο username και password, να φτιάξουμε τα απαραίτητα views και να δώσουμε τα permissions που του αντιστοιχούν). Μπορείτε τώρα να αρχίσετε να χρησιμοποιείτε τη βάση δεδομένων είτε ως admin, είτε ως chef user και να εκτελέσετε τα ερωτήματα που επιθυμείτε. Για να αποσυνδεθείτε πληκτρολογήστε στο τερματικό exit.

5. Ανάλυση λογικής queries

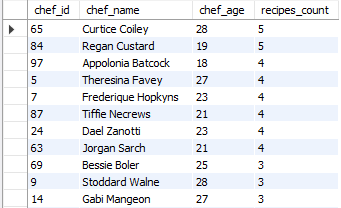
1. Σε αυτό το query για να υπολογίσουμε τον μέσο όρο των αξιολογήσεων κάθε σεφ και κάθε εθνικής κουζίνας ακολουθήσαμε 2 φορές την ίδια διαδικασία. Συγκεκριμένα δημιουργήσαμε 2 φορές μια απλή δομή select, η οποία επιστρέφει ένα πίνακα που περιέχει τον μέσο όσο των συνολικών βαθμών (με την χρήση της εντολής avg) και το αναγνωριστικό του μάγειρα ή της εθνικής κουζίνας. Παρακάτω παρατίθεται ενδεικτικά ένα απόσπασμα του αποτελέσματος.



1. Σε αυτό το query ορίσαμε αυθαίρετα μια τυχαία εθνική κουζίνα και μια σεζόν ώστε να εμφανιστούν ενδεικτικά αποτελέσματα. Αρχικά βρήκαμε τους μάγειρες που ανήκουν σε αυτήν την εθνική κουζίνα. Έπειτα δημιουργήσαμε έναν μεγάλο πίνακα, ο οποίος περιείχε χαρακτηριστικά από τους πίνακες ethnic\_of\_chef, ethnic\_cuisines, episode\_information και recipe, οι οποίοι ενώθηκαν με πολλά join στα κοινά τους χαρακτηριστικά, οπότε βρήκαμε μάγειρες που ανήκουν στην εθνική κουζίνα Mexican και εμφανίζονται σε κάποιο επεισόδιο της σεζόν 1. Παρακάτω παρατίθεται ενδεικτικά ένα απόσπασμα του αποτελέσματος.



1. Το αποτέλεσμα που δίνει αυτό το query είναι ένας πίνακας που περιέχει τα χαρακτηριστικά chef\_name, chef\_age ,recipes\_count. Για να γίνει αυτό χρησιμοποιήσαμε και πάλι την εντολή count αλλά και τις εντολές sum για να βρούμε το άθροισμα των συνταγών και την εντολή concat ώστε να εκτυπώσουμε αποτέλεσμα με συγκεκριμένη μορφή. Παρακάτω παρατίθεται ενδεικτικά ένα απόσπασμα του αποτελέσματος.



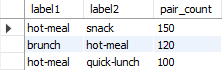
1. Η διαφορά αυτού του query από τα προηγούμενα είναι πως δεν θέλουμε να βρούμε εγγραφές οι οποίες θα έχουν ένα χαρακτηριστικό αλλά εγγραφές που δεν θα έχουν αυτό το χαρακτηριστικό. Για αυτό τον σκοπό χρησιμοποιούμε την εντολή where not exists μέσα στο query μας. Παρακάτω παρατίθεται ενδεικτικά ένα απόσπασμα του αποτελέσματος.



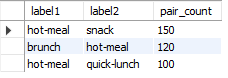
1. Η ιδιαιτερότητα αυτού του query έγκειται στο γεγονός ότι κάνουμε ένωση ενός πίνακα με τον εαυτό του. Με αυτό τον τρόπο βρίσκουμε 2 κριτές του episode\_information οι οποίοι στην ίδια σεζόν(έστω στην δεύτερη) έχουν τον ίδιο αριθμό συμμετοχών(>3) ως κριτές. Για να μην μετρήσουμε 2 φορές το κάθε συνδυασμό κριτών έχουμε θέσει τον περιορισμό το id του πρώτου να είναι πάντα μικρότερο του δεύτερου.



1. Αυτό το query εντοπίζει και μετρά τα πιο συχνά ζευγάρια ετικετών που εμφανίζονται μαζί σε συνταγές. Συγκεκριμένα, χρησιμοποιεί δύο ψευδώνυμα για τον πίνακα labels\_of\_recipe (l1,l2), τα οποία συνδέει με βάση το recipe\_id, επιτρέποντας τον εντοπισμό ζευγαριών ετικετών που ανήκουν στην ίδια συνταγή. Αφού εξασφαλίσουμε ότι κάθε ζευγάρι εμφανίζεται μία φορά, στην συνέχεια φροντίζουμε να συνδέσουμε το query με τον πίνακα των επεισοδίων episodes\_infromation(ei) με βάση το recipe\_id, για να εξασφαλιστεί η εγκυρότητα των συνταγών. Έπειτα, μέσω της εντολήσ GROUP\_BY και της συνάρτησης COUNT\*, επιτυγχάνουμε την ομαδοποίηση και την μέτρηση αντίστοιχα. Στην συνέχεια ,ταξινομούμε και περιορίζουμε τα αποτελέσματα.



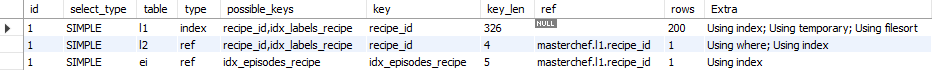
**Εναλλακτικό Query Plan:** Στο σημείο αυτό χρησιμοποιούμε το ίδιο query με την διαφορά ότι προσθέτουμε την εντολή FORCE INDEX, που δηλώνει το index που μας βοηθά στον υπολογισμό του ερωτήματος. Εκτελούμε το query και παίρνουμε το ίδιο αποτέλεσμα:

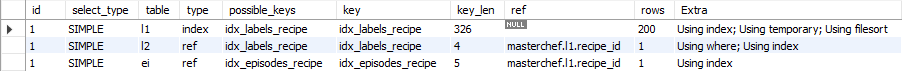


Όταν εκτελέσουμε το EXPLAIN για τα παραπάνω queries, θα πάρουμε έναν πίνακα που περιγράφει τον τρόπο με τον οποίο το MySQL διαχειρίζεται το query. Τα κύρια πεδία που πρέπει να προσέξουμε είναι:

* table: Ο πίνακας που εξετάζεται σε αυτό το βήμα.
* type: Ο τύπος της ένωσης (JOIN) που χρησιμοποιείται.
* possible\_keys: Τα ευρετήρια που θα μπορούσαν να χρησιμοποιηθούν για αυτό το query.
* key: Το ευρετήριο που χρησιμοποιείται τελικά.
* key\_len: Το μήκος του χρησιμοποιούμενου ευρετηρίου.
* ref: Οι στήλες ή οι σταθερές που χρησιμοποιούνται με το ευρετήριο.
* rows: Ο εκτιμώμενος αριθμός των γραμμών που θα διαβαστούν από τον πίνακα.
* filtered: Το ποσοστό των γραμμών που αναμένεται να περάσουν το φίλτρο.
* Extra: Πρόσθετες πληροφορίες, όπως αν χρησιμοποιείται προσωρινός πίνακας ή αρχειοθέτηση.

Προκύπτουν λοιπόν τα παρακάτω αποτελέσματα για το απλό query και το query με force indexes αντίστοιχα:



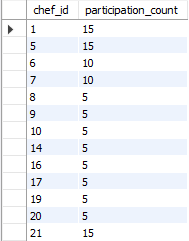


Και στα δύο queries, ο αριθμός των γραμμών που θα διαβαστούν είναι ο ίδιος (200 για τον πίνακα l1, 1 για τους πίνακες l2 και ei). Το απλό query μπορεί να χρησιμοποιήσει περισσότερα ευρετήρια, ενώ το query με force index χρησιμοποιεί μόνο συγκεκριμένα ευρετήρια, όμως φαίνεται να έχουν παρόμοια αποδοτικότητα. Για να έχουμε μία πιο ακριβή εικόνα σχετικά με το χρόνο εκτέλεσης του κάθε query εκτελούμε την εντολή *set profiling = 1*; και στην συνέχεια εκτελόυμε τα δύο queries. Αν χρησιμοποιήσουμε την εντολή *show profiles;* παίρνουμε τοΝ παρακάτω πίνακα.



Παρατηρούμε δηλαδή ότι με force index το query εκτελείται γρηγορότερα.

1. Aυτό το SQL ερώτημα εντοπίζει τους σεφ που έχουν συμμετάσχει τουλάχιστον 5 λιγότερες φορές από τον μάγειρα με τις περισσότερες συμμετοχές σε επεισόδια. Για τους σεφ όπου ο ρόλος είναι 1 επιλέγεται το chef\_id και ο αριθμός συμμετοχών count από τον πίνακα επεισοδίων. Το υποερώτημα subquery υπολογίζει τον μέγιστο αριθμό συμμετοχών MAX(participation\_count)όλων των σεφ αφαιρώντας 5. Στην συνέχεια, μέσω του φίλτρου HAVING φιλτράρονται τα αποτελέσματα ώστε να περιλαμβάνει μόνο τους σεφ των οποίων ο αριθμός συμμετοχών είναι μικρότερος η ίσος με τον μέγιστο αριθμό συμμετοχών μείον 5. Βρίσκουμε (με το δεύτερο query που έχουμε φτιάξει) ότι ο μάγειρας με το μεγαλύτερο αριθμό συμμετοχών είναι αυτός που αντιστοιχεί στο chef\_id=4 με 25 εμφανίσεις. Παρακάτω παρατίθεται ενδεικτικά ένα απόσπασμα του αποτελέσματος.



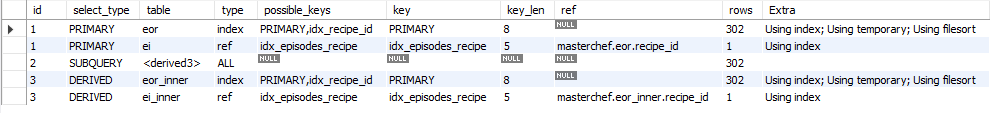
1. Αυτό το ερώτημα βρίσκει τα επεισόδια που χρησιμοποίησαν τον μέγιστο αριθμό εξοπλισμού και τα επιστρέφει ταξινομημένα κατά episode\_id. Ειδικότερα, ενώνει τον πίνακα episodes\_infromation (ei) με τον πίνακα equipment\_of\_recipe (eor) με γνώμονα το recipe και το recipe\_id αντίστοιχα. Ακόμα, ομαδοποίει τα αποτελέσματα ανά episode\_id και μετρά τον αριθμό εξοπλισμού που χρησιμοποιήθηκε σε κάθε επεισόδιο. Το φίλτρο HAVING διασφαλίζει ότι μόνο τα επεισόδια μέγιστου αριθμού εξοπλισμού συμπεριλαμβάνονται στο αποτέλεσμα. Αυτό επιτυγχάνεται με ένα υποερώτημα που υπολογίζει τον μέγιστο αριθμό εξοπλισμού που χρησιμοποιήθηκε σε οποιοδήποτε επεισόδιο. Τέλος, τα αποτελέσματα ταξινομούνται κατά episode\_id.

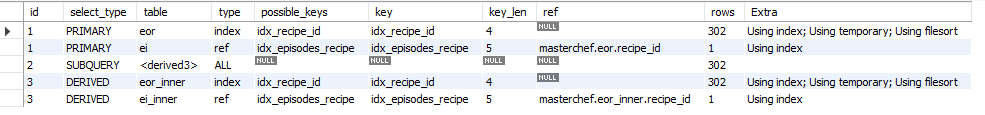


**Ενναλακτικό Query Plan:** Στο σημείο αυτό βάζουμε force index .Tο idx\_recipe\_id για τον πίνακα recipe\_id επιτυγχάνει την αναζήτηση και την ένωση βάση συνταγής, ενώ το idx\_episodes\_recipe για τον πίνακα episodes\_infromation για την στήλη recipe επιτυχάνει επίσης την αναζήτηση και την ένωση βάση συνταγής.



Η χρήση της force index επιβάλλει μάλιστα στο ερώτημα να χρησιμοποιήσει τα ανώτερα ευρετήρια. Τρέχοντας την εντολή explain για τα δύο queries λαμβάνουμε τα παρακάτω αποτελέσματα:



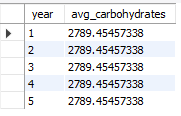


Από την ανάλυση των EXPLAIN αποτελεσμάτων, παρατηρούμε ότι και στα δύο queries, τα indexes που χρησιμοποιούνται είναι τα ίδια. Αν εκτελέσουμε τις εντολές *set profiling = 1*; και *show profiles;* παίρνουμε τα παρακάτω αποτελέσματα σχετικά με τον χρόνο εκτέλεσης.

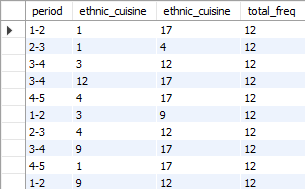


Παρατηρούμε δηλαδή ότι το απλό query εκτελείται γρηγορότερα. Το force index άρα μας εξυπηρετεί κυρίως, ώστε να επιλέξουμε ένα συγκεκριμένο μονοπάτι, αλλά από άποψη χρόνου δεν προσφέρει πάντα βελτίωση.

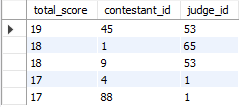
1. Αυτό το query, αρχικά συνδέει τον πίνακα episodes\_infromation (ei) με το ingredients\_of\_recipe (ir) με βάση το recipe\_id, καθώς και τον ingredients\_of\_recipe (ir) με τον ingredients(i) με βάση το ingredient\_id. Στην συνέχεια υπολογίζουμε τον μέσο όρο υδατανθράκων για κάθε σεζόν. Τέλος, τα αποτελέσματα ομαδοποιούνται και ταξινομούνται κατά season\_of\_episode και έτσι λαμβάνουμε το επιθυμητό αποτέλεσμα.



1. Η ιδιαιτερότητα της λύσης που έχουμε δώσει σε αυτό το query είναι πως με την εντολή with δημιουργούμε έναν προσωρινό πίνακα τον οποίον στην συνέχεια χρησιμοποιούμε εντός της with ώστε να δημιουργήσουμε έναν νέο πίνακα. Αυτός ο πίνακας περιέχει την συχνότητα εμφάνισης των εθνικών κουζινών. Κάνοντας join για 2 στοιχεία αυτού του πίνακα βρίσκουμε τις εθνικές κουζίνες με ίδια συχνότητα εμφάνισης σε διάστημα 2 συνεχόμενων ετών. Δίνεται ενδεικτικά ένα απόσπασμα του αποτελέσματος.



1. Στο query αυτό υπολογίσαμε το άθροισμα των βαθμολογιών που έχουν δώσει όλοι οι κριτές σε κάθε μάγειρα και αποθηκεύσαμε τις πέντε μεγαλύτερες βαθμολογίες. Για να βρούμε ποιος κριτής έδωσε τον κάθε βαθμό στους μάγειρες ακολουθούμε τον επόμενο κώδικα. Όταν το role\_of-chef=0,δηλαδή ο συγκεκριμένος chef είναι κριτής, θέτουμε 0 τον i-οστο βαθμό και NULL τους άλλους 2. Για παράδειγμα αν κάποιος κριτής είναι ο δεύτερος κριτής ενός επεισοδίου θα έχει τις τιμές (grade1,grade2,grade3)=(NULL,0,NULL). Όσον αφορά τον sql κώδικα χρησιμοποιούμε την εντολή case για να βρούμε τους κριτές.



1. Η υλοποίηση αυτού του query γίνεται σε τρία μέρη. Αρχικά, μέσω της εντολής AVG εντός ενός with υπολογίζουμε έναν πίνακα με την μέση δυσκολία των επεισοδίων ανά έτος. Στην συνέχεια, βρίσκουμε το επεισόδιο με την μέγιστη μέση δυσκολία και αποθηκεύουμε την τιμή της δυσκολίας στο max-average-difficulty. Τέλος, με βάση την τιμή που έχουμε βρει βρίσκουμε από στον πίνακα με τις μέσες δυσκολίες τα χαρακτηριστικά του επεισοδίου που αντιστοιχεί.



1. Το ιδιαίτερο χαρακτηριστικό αυτού του query είναι πως πρέπει να αντιστοιχίσουμε τις επιμέρους τιμές επαγγελματικής κατάρτισης σε ακέραιους αριθμούς ώστε να κάνουμε την πρόσθεση των ικανοτήτων του κάθε chef. Πέρα από αυτό, χρησιμοποιούμε και πάλι την εντολή with και την εντολή min ώστε να υπολογίσουμε την ελάχιστη συνολική τιμή επαγγελματικής κατάρτισης ανά επεισόδιο.



1. Η απάντηση σε αυτό το query όπως και στο query 12 δίνεται σε 3 μέρη. Με παρόμοια λογική στην αρχή υπολογίζουμε τις εμφανίσεις της κάθε θεματικής ενότητας στον διαγωνισμό, στην συνέχεια τον μέγιστο αριθμό εμφανίσεων και εν τέλει τον θεματική ενότητα στην οποία αντιστοιχεί αυτή η μέγιστη τιμή.



1. Αυτό το query είναι παρόμοιο με το query 4 αφού πάλι θέλουμε να βρούμε εγγραφές που να μην πληρούν κάποιο κριτήριο. Για τον σκοπό αυτό χρησιμοποιούμε την εντολή where <attribute> not in και βρίσκουμε την ομάδα τροφίμων που θέλουμε στο κατά τα άλλα εύκολο query.



6. Σύνδεσμος github repository

Ο συνολικός κώδικας της εφαρμογής μας περιέχεται στο παρακάτω github repository:

<https://github.com/Madeperf/DataBaseProject>