

Εθνικό Μετσόβιο Πολυτεχνείο



Βάσεις Δεδομένων Αναφορά Εξαμηνιαίας Εργασίας

Ομάδα 15

Αντωνία Πάπαλου - ge20706 - papalouantonio@gmail.com

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

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

Μάιος 2024

Περιεχόμενα

Εισαγωγή	3
1. Διάγραμμα ER (Entity-Relation Diagram)	4
2. Σχεσιακό Διάγραμμα (Schema Diagram)	5
3. Περιορισμοί (Constraints) Βάσης Δεδομένων	6
4. Triggers	10
5. Stored Procedures	11
6. Indexes	11
7. DDL script	11
8. DML script	18
Views	19
9. Random Episode Generator	19
10. Permissions	24
11. Queries-Ερωτήματα	26
11.1	26
Εκφώνηση:	26
Κώδικας:	27
Αποτελέσματα:	27
11.2	27
Εκφώνηση:	27
Παραδοχές:	27
Κώδικας:	27
Αποτελέσματα:	28
11.3	28
Εκφώνηση:	28
Κώδικας:	28
Αποτελέσματα:	28
11.4	29
Εκφώνηση:	29
Κώδικας:	29
Αποτελέσματα:	29
11.5	29
Εκφώνηση:	29
Κώδικας:	29
Αποτελέσματα:	30
11.6	30

	Εκφώνηση:	30
	Παραδοχές:	30
	Κώδικας:	30
	Αποτελέσματα:	31
11.7		31
	Εκφώνηση:	31
	Κώδικας:	31
	Αποτελέσματα:	32
11.8		32
	Εκφώνηση:	32
	Παραδοχές:	32
	Κώδικας:	33
	Αποτελέσματα:	33
11.9		33
	Εκφώνηση:	33
	Παραδοχές:	34
	Κώδικας:	34
	Αποτελέσματα:	34
11.10		34
	Εκφώνηση:	34
	Κώδικας:	34
	Αποτελέσματα:	35
11.11		35
	Εκφώνηση:	35
	Κώδικας:	35
	Αποτελέσματα:	36
11.12		36
	Εκφώνηση:	36
	Κώδικας:	36
	Αποτελέσματα:	36
11.13		37
	Εκφώνηση:	37
	Κώδικας:	37
	Αποτελέσματα:	37
11.14		37
	Εκφώνηση:	37
	Κώδικας:	37
	Αποτελέσματα:	38
11.15		38
	Εκφώνηση:	38
	Παραδοχές:	38
	Κώδικας:	38
	Αποτελέσματα:	38
12. Οδηγίες Εγκατάστασης Βάσης		39

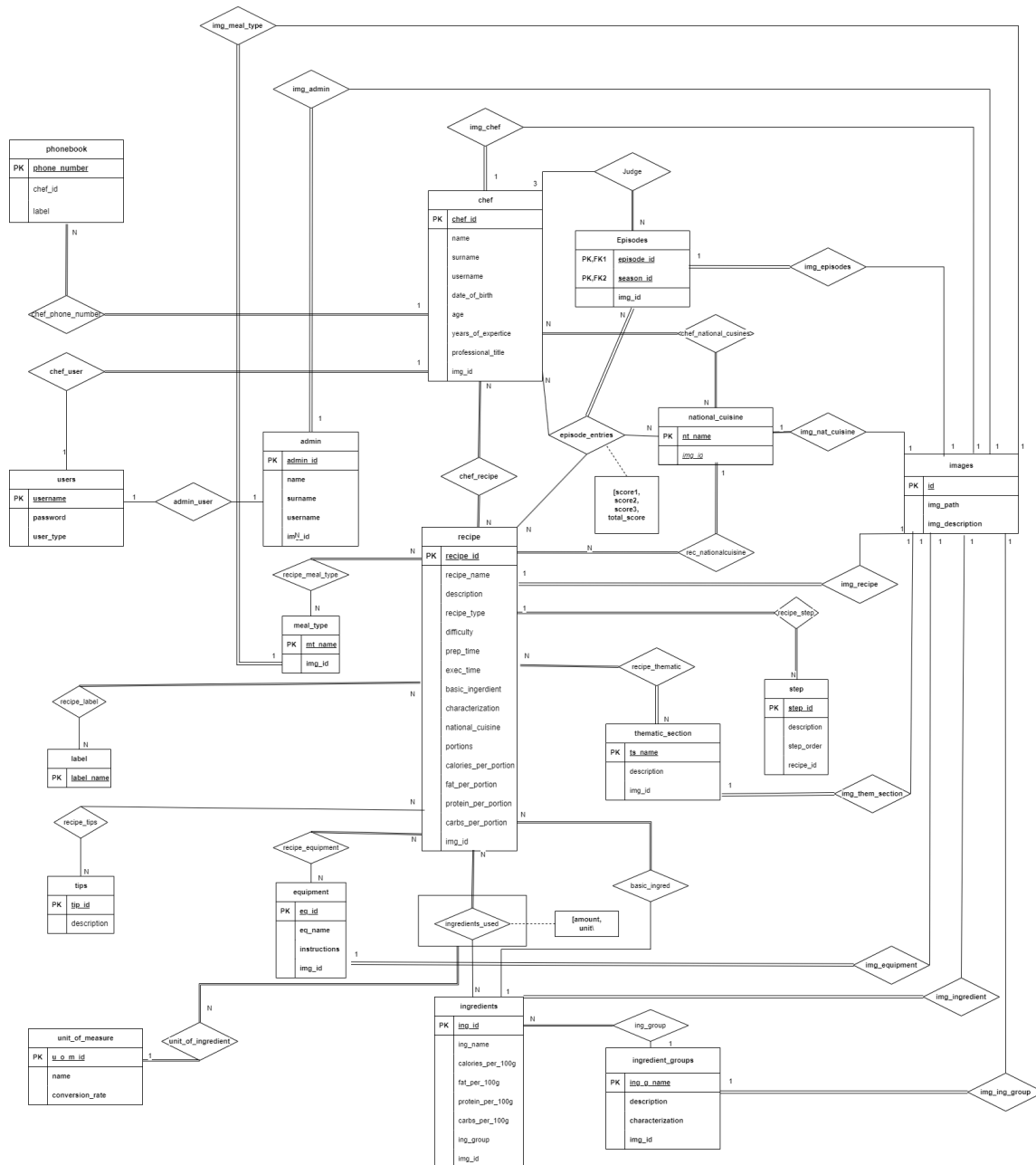
Εισαγωγή

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

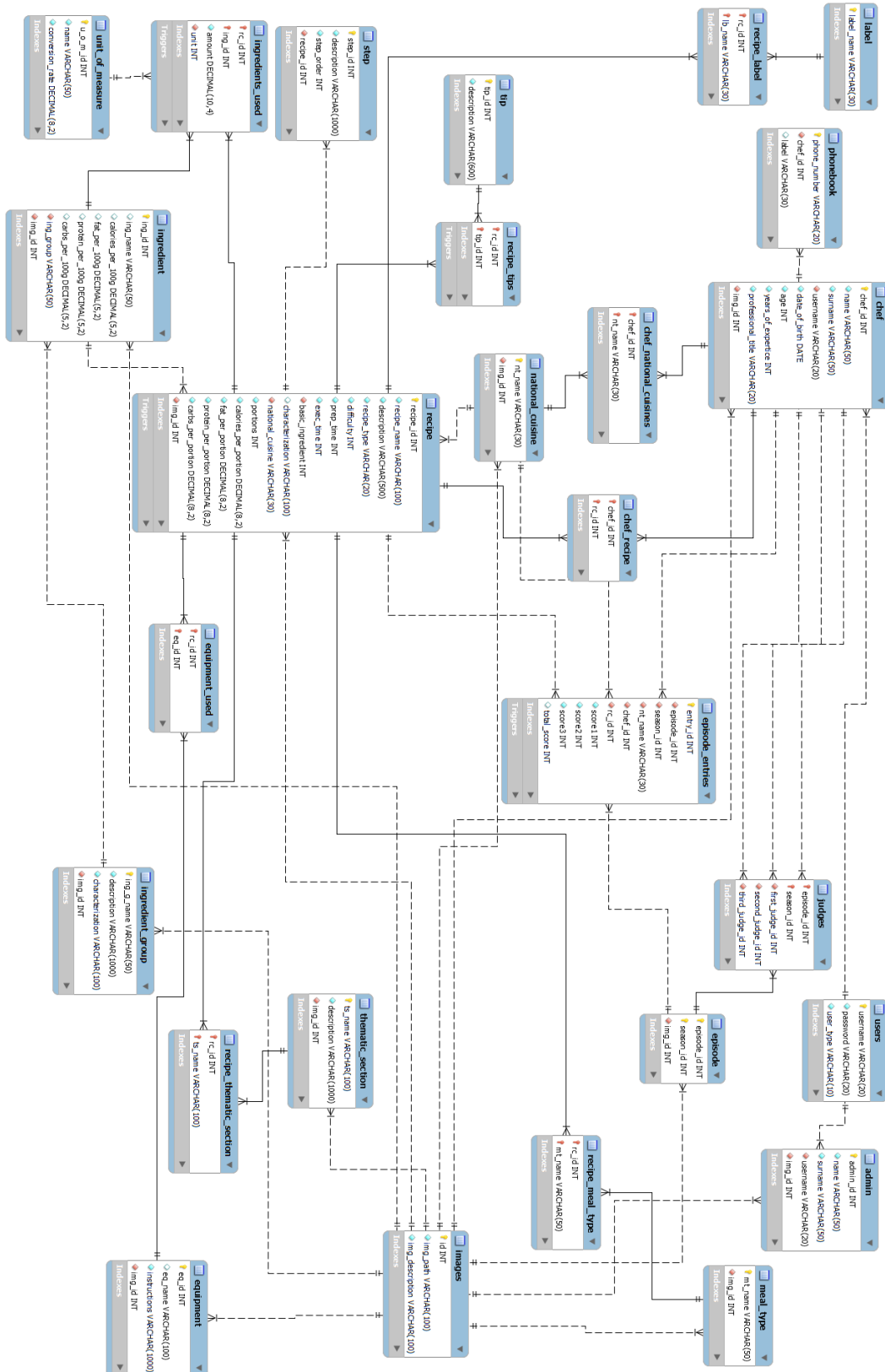
Η αναφορά αυτή περιέχει τη διαδικασία ανάπτυξης της εν λόγω βάσης, τους κώδικες για την κατασκευή της βάσης και την εισαγωγή δεδομένων σε αυτή, τους κώδικες για την κλήρωση τυχαίων επεισοδίων βάση κάποιων περιορισμών καθώς και κάποια ερωτήματα(queries) που κληθήκαμε να απαντήσουμε σε SQL.

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

1. Διάγραμμα ER (Entity-Relation Diagram)



2. Σχεσιακό Διάγραμμα (Schema Diagram)



3. Περιορισμοί (Constraints) Βάσης Δεδομένων

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

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

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

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

- Στον πίνακα judges έχουμε δευτερεύον κλειδί τον συνδυασμό των στηλών episode_id και season_id που αναφέρεται στις στήλες episode_id και season_id του πίνακα episode
- Στον πίνακα judges έχουμε δευτερεύον κλειδί τη στήλη first_judge_id που αναφέρεται στη στήλη chef_id του πίνακα chef
- Στον πίνακα judges έχουμε δευτερεύον κλειδί τη στήλη second_judge_id που αναφέρεται στη στήλη chef_id του πίνακα chef
- Στον πίνακα judges έχουμε δευτερεύον κλειδί τη στήλη third_judge_id που αναφέρεται στη στήλη chef_id του πίνακα chef
- Unique Constraints : Επιβάλλει μοναδικότητα τιμών σε στήλη ή στήλες, επιτρέπει μία τιμή NULL.
 - Στο πίνακα tip η στήλη description έχει τον περιορισμό Unique αφού δεν επιτρέπουμε σε δύο διαφορετικά tip να έχουν την ίδια περιγραφή.
- Not null Constraints : Απαγορεύει τις NULL τιμές σε μια στήλη. Σχεδόν όλες οι στήλες παίρνουν αυτό τον περιορισμό ώστε να μην υπάρχουν κενά στα δεδομένα μας.
- Check Constraints : Επιβάλλει συγκεκριμένες συνθήκες στις τιμές μιας στήλης.
 - Στο πίνακα ingredient υπάρχουν περιορισμοί Check στις στήλες calories_per_100g, fat_per_100g, protein_per_100g, carbs_per_100g που διασφαλίζουν την ορθότητα των τιμών των στηλών. Επίσης υπάρχει ακόμα ένας check περιορισμός με όνομα sum_of_macros το οποίο διασφαλίζει ότι το άθροισμα των macros ενός συστατικού δεν ξεπερνά τα 100 γραμμάρια.
 - Στον πίνακα users υπάρχει ένας check περιορισμός με όνομα password_length που ελέγχει αν τα δεδομένα της στήλης password έχουν το σωστό μήκος και ένας check περιορισμός με όνομα user_type_accepted_values που ελέγχει αν οι τιμές της στήλης user_type είναι οι επιτρεπτές.
 - Στον πίνακα users υπάρχει ένας check περιορισμός στις στήλες age και year_of_expertice που διασφαλίζει ότι είναι θετικοί αριθμοί.
 - Στον πίνακα unit_of_measure υπάρχει ένας check περιορισμός στη στήλη conversion_rate που διασφαλίζει ότι είναι θετικός αριθμός.
 - Στον πίνακα recipe υπάρχουν check περιορισμοί στις στήλες difficulty, prep_time, exec_time, portions, calories_per_portion, fat_per_portion, protein_per_portion, carbs_per_portion που διασφαλίζουν ότι οι στήλες παίρνουν έγκυρες τιμές. Επίσης υπάρχει ακόμα ένας check περιορισμός με όνομα recipe_type_values το οποίο διασφαλίζει ότι το η στήλη recipe_type παίρνει επιτρεπτές τιμές.
 - Στον πίνακα ingredients_used υπάρχει ένας check περιορισμός στη στήλη amount που διασφαλίζει ότι είναι θετικός αριθμός.
 - Στον πίνακα episode_entries υπάρχουν check περιορισμοί στις στήλες score1, score2, score3 που διασφαλίζουν ότι παίρνουν τιμές από 1 μέχρι 5.
- Default : Ορίζει μια προεπιλεγμένη τιμή για μια στήλη αν δεν δοθεί άλλη τιμή.

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

4. Triggers

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

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

5. Stored Procedures

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

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

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

6. Indexes

Τα indexes (ευρετήρια) χρησιμοποιούνται από το rdbms για ταχύτερη εύρεση εγγραφών, οπότε είναι χρήσιμα για την ταχύτερη λειτουργία της βάσης δεδομένων σε διάφορα queries. Η MySQL εισάγει αυτόματα indexes για όλα τα πρωτεύοντα και δευτερεύοντα κλειδιά των πινάκων, επομένως οι πλήστες από τις στήλες που χρησιμοποιούνται στα ερωτήματα είχαν ήδη ευρετήριο και δεν χρειάστηκε να δημιουργήσουμε εμείς.

Η μόνη στήλη στην οποία δημιουργήσαμε index είναι η στήλη age του πίνακα chef έτσι ώστε να τρέχει ταχύτερα το ερώτημα 3.3

7. DDL script

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

```
1 DROP DATABASE IF EXISTS cooking_competition;
2
3 CREATE DATABASE cooking_competition;
4 USE cooking_competition;
5
6 /*-----TABLES-----*/
7 CREATE TABLE images (
8     id INT AUTO_INCREMENT PRIMARY KEY,
9     img_path VARCHAR(100) NOT NULL,
10    img_description VARCHAR(100) NOT NULL
```

```

11 );
12 CREATE TABLE meal_type (
13     mt_name VARCHAR(50) PRIMARY KEY,
14     img_id INT NOT NULL,
15     FOREIGN KEY (img_id) REFERENCES images(id) ON DELETE RESTRICT ON UPDATE CASCADE
16 );
17 CREATE TABLE label (
18     label_name VARCHAR(30) PRIMARY KEY
19 );
20 CREATE TABLE tip (
21     tip_id INT AUTO_INCREMENT PRIMARY KEY,
22     description VARCHAR(600) UNIQUE NOT NULL
23 );
24 CREATE TABLE equipment(
25     eq_id INT AUTO_INCREMENT PRIMARY KEY,
26     eq_name VARCHAR(100) NOT NULL,
27     instructions VARCHAR(1000) NOT NULL,
28     img_id INT NOT NULL,
29     FOREIGN KEY (img_id) REFERENCES images(id) ON DELETE RESTRICT ON UPDATE CASCADE
30 );
31 CREATE TABLE ingredient_group(
32     ing_g_name VARCHAR(50) PRIMARY KEY,
33     description VARCHAR(1000) NOT NULL,
34     characterization VARCHAR(100) NOT NULL,
35     img_id INT NOT NULL,
36     FOREIGN KEY (img_id) REFERENCES images(id) ON DELETE RESTRICT ON UPDATE CASCADE
37 );
38 CREATE TABLE ingredient(
39     ing_id INT AUTO_INCREMENT PRIMARY KEY,
40     ing_name VARCHAR(50) NOT NULL,
41     calories_per_100g NUMERIC(5,2) CHECK(calories_per_100g >= 0 AND calories_per_100g<=900),
42     fat_per_100g NUMERIC(5,2) CHECK(fat_per_100g >= 0 AND fat_per_100g <= 100),
43     protein_per_100g NUMERIC(5,2) CHECK(protein_per_100g >= 0 AND protein_per_100g <= 100),
44     carbs_per_100g NUMERIC(5,2) CHECK(carbs_per_100g >= 0 AND carbs_per_100g <= 100),
45     ing_group VARCHAR(50) NOT NULL,
46     img_id INT NOT NULL,
47     FOREIGN KEY (img_id) REFERENCES images(id) ON DELETE RESTRICT ON UPDATE CASCADE,
48     FOREIGN KEY (ing_group) REFERENCES ingredient_group(ing_g_name) ON DELETE RESTRICT ON UPDATE CASCADE,
49     CONSTRAINT sum_of_macros CHECK (fat_per_100g+protein_per_100g+carbs_per_100g <= 100)
50 );
51 CREATE TABLE thematic_section(
52     ts_name VARCHAR(100) PRIMARY KEY,
53     description VARCHAR(1000) NOT NULL,
54     img_id INT NOT NULL,
55     FOREIGN KEY (img_id) REFERENCES images(id) ON DELETE RESTRICT ON UPDATE CASCADE
56 );
57 CREATE TABLE national_cuisine(
58     nt_name VARCHAR(30) PRIMARY KEY,
59     img_id INT NOT NULL,
60     FOREIGN KEY (img_id) REFERENCES images(id) ON DELETE RESTRICT ON UPDATE CASCADE
61 );
62 CREATE TABLE users(
63     username VARCHAR(20) PRIMARY KEY,
64     password VARCHAR(20) NOT NULL,
65     user_type VARCHAR(10) NOT NULL,
66     CONSTRAINT password_length CHECK (LENGTH(password) BETWEEN 7 AND 21),
67     CONSTRAINT user_type_accepted_values CHECK (user_type in ('admin','chef'))
68 );
69 CREATE TABLE chef(
70     chef_id INT PRIMARY KEY,
71     name VARCHAR(50) NOT NULL,
72     surname VARCHAR(50) NOT NULL,
73     username VARCHAR(20) NOT NULL,
74     date_of_birth DATE NOT NULL,
75     age INT NOT NULL CHECK (age > 0),
76     years_of_expertise INT NOT NULL CHECK (years_of_expertise >= 0),
77     professional_title VARCHAR(20) NOT NULL,
78     img_id INT NOT NULL,
79     FOREIGN KEY (img_id) REFERENCES images(id) ON DELETE RESTRICT ON UPDATE CASCADE,

```

```

80 FOREIGN KEY (username) REFERENCES users(username) ON DELETE RESTRICT ON UPDATE CASCADE
81 );
82 CREATE TABLE admin(
83     admin_id INT PRIMARY KEY,
84     name VARCHAR(50) NOT NULL,
85     surname VARCHAR(50) NOT NULL,
86     username VARCHAR(20) NOT NULL,
87     img_id INT NOT NULL,
88     FOREIGN KEY (img_id) REFERENCES images(id) ON DELETE RESTRICT ON UPDATE CASCADE,
89     FOREIGN KEY (username) REFERENCES users(username) ON DELETE RESTRICT ON UPDATE CASCADE
90 );
91 CREATE TABLE phonebook(
92     phone_number VARCHAR(20) PRIMARY KEY,
93     chef_id INT NOT NULL,
94     label VARCHAR(30),
95     FOREIGN KEY (chef_id) REFERENCES chef(chef_id) ON DELETE RESTRICT ON UPDATE CASCADE
96 );
97 CREATE TABLE unit_of_measure(
98     u_o_m_id INT PRIMARY KEY,
99     name VARCHAR(50) NOT NULL,
100     conversion_rate NUMERIC(8,2) NOT NULL CHECK (conversion_rate >= 0)
101 );
102 CREATE TABLE recipe(
103     recipe_id INT AUTO_INCREMENT PRIMARY KEY,
104     recipe_name VARCHAR(100) NOT NULL,
105     description VARCHAR(500) NOT NULL,
106     recipe_type VARCHAR(20) NOT NULL,
107     difficulty INT NOT NULL CHECK(difficulty >= 1 AND difficulty <= 5),
108     prep_time INT NOT NULL CHECK(preptime >= 0),
109     exec_time INT NOT NULL CHECK(exec_time >= 0),
110     basic_ingredient INT NOT NULL,
111     characterization VARCHAR(100) ,
112     national_cuisine VARCHAR(30) NOT NULL,
113     portions INT NOT NULL CHECK(portions > 0 ) DEFAULT 0,
114     calories_per_portion NUMERIC(8,2) NOT NULL CHECK (calories_per_portion >= 0) DEFAULT 0,
115     fat_per_portion NUMERIC(8,2) NOT NULL CHECK (fat_per_portion >= 0) DEFAULT 0,
116     protein_per_portion NUMERIC(8,2) NOT NULL CHECK (protein_per_portion >= 0) DEFAULT 0,
117     carbs_per_portion NUMERIC(8,2) NOT NULL CHECK (carbs_per_portion >= 0) DEFAULT 0,
118     img_id INT NOT NULL,
119     FOREIGN KEY (img_id) REFERENCES images(id) ON DELETE RESTRICT ON UPDATE CASCADE,
120     FOREIGN KEY (national_cuisine) REFERENCES national_cuisine(nt_name) ON DELETE RESTRICT ON UPDATE CASCADE,
121     FOREIGN KEY (basic_ingredient) REFERENCES ingredient(ing_id) ON DELETE RESTRICT ON UPDATE CASCADE,
122     CONSTRAINT recipe_type_values CHECK (recipe_type in ('savory','confectionery'))
123 );
124 CREATE TABLE step(
125     step_id INT AUTO_INCREMENT PRIMARY KEY,
126     description VARCHAR(1000) NOT NULL,
127     step_order INT NOT NULL,
128     recipe_id INT NOT NULL,
129     FOREIGN KEY (recipe_id) REFERENCES recipe(recipe_id) ON DELETE RESTRICT ON UPDATE CASCADE
130 );
131 CREATE TABLE recipe_meal_type(
132     rc_id INT NOT NULL,
133     mt_name VARCHAR(50) NOT NULL,
134     PRIMARY KEY (rc_id,mt_name),
135     FOREIGN KEY (rc_id) REFERENCES recipe(recipe_id) ON DELETE RESTRICT ON UPDATE CASCADE,
136     FOREIGN KEY (mt_name) REFERENCES meal_type(mt_name) ON DELETE RESTRICT ON UPDATE CASCADE
137 );
138 CREATE TABLE recipe_label(
139     rc_id INT NOT NULL,
140     lb_name VARCHAR(30) NOT NULL,
141     PRIMARY KEY (rc_id,lb_name),
142     FOREIGN KEY (rc_id) REFERENCES recipe(recipe_id) ON DELETE RESTRICT ON UPDATE CASCADE,
143     FOREIGN KEY (lb_name) REFERENCES label(label_name) ON DELETE RESTRICT ON UPDATE CASCADE
144 );
145 CREATE TABLE recipe_tips(
146     rc_id INT NOT NULL,
147     tip_id INT NOT NULL,
148     PRIMARY KEY (rc_id,tip_id),

```

```

149     FOREIGN KEY (rc_id) REFERENCES recipe(recipe_id) ON DELETE RESTRICT ON UPDATE CASCADE,
150     FOREIGN KEY (tip_id) REFERENCES tip(tip_id) ON DELETE RESTRICT ON UPDATE CASCADE
151 );
152 CREATE TABLE equipment_used(
153     rc_id INT NOT NULL,
154     eq_id INT NOT NULL,
155     PRIMARY KEY (rc_id,eq_id),
156     FOREIGN KEY (rc_id) REFERENCES recipe(recipe_id) ON DELETE RESTRICT ON UPDATE CASCADE,
157     FOREIGN KEY (eq_id) REFERENCES equipment(eq_id) ON DELETE RESTRICT ON UPDATE CASCADE
158 );
159 CREATE TABLE ingredients_used(
160     rc_id INT NOT NULL,
161     ing_id INT NOT NULL,
162     amount NUMERIC(10,4) NOT NULL CHECK(amount > 0),
163     unit INT NOT NULL,
164     PRIMARY KEY (rc_id,ing_id),
165     FOREIGN KEY (rc_id) REFERENCES recipe(recipe_id) ON DELETE RESTRICT ON UPDATE CASCADE,
166     FOREIGN KEY (unit) REFERENCES unit_of_measure(u_o_m_id) ON DELETE RESTRICT ON UPDATE CASCADE,
167     FOREIGN KEY (ing_id) REFERENCES ingredient(ing_id) ON DELETE RESTRICT ON UPDATE CASCADE
168 );
169 CREATE TABLE recipe_thematic_section(
170     rc_id INT NOT NULL,
171     ts_name VARCHAR(100) NOT NULL,
172     PRIMARY KEY (rc_id,ts_name),
173     FOREIGN KEY (rc_id) REFERENCES recipe(recipe_id) ON DELETE RESTRICT ON UPDATE CASCADE,
174     FOREIGN KEY (ts_name) REFERENCES thematic_section(ts_name) ON DELETE RESTRICT ON UPDATE CASCADE
175 );
176 CREATE TABLE chef_national_cuisines(
177     chef_id INT NOT NULL,
178     nt_name VARCHAR(30) NOT NULL,
179     PRIMARY KEY (chef_id,nt_name),
180     FOREIGN KEY (chef_id) REFERENCES chef(chef_id) ON DELETE RESTRICT ON UPDATE CASCADE,
181     FOREIGN KEY (nt_name) REFERENCES national_cuisine(nt_name) ON DELETE RESTRICT ON UPDATE CASCADE
182 );
183 CREATE TABLE chef_recipe(
184     chef_id INT NOT NULL,
185     rc_id INT NOT NULL,
186     PRIMARY KEY (chef_id,rc_id),
187     FOREIGN KEY (chef_id) REFERENCES chef(chef_id) ON DELETE RESTRICT ON UPDATE CASCADE,
188     FOREIGN KEY (rc_id) REFERENCES recipe(recipe_id) ON DELETE RESTRICT ON UPDATE CASCADE
189 );
190 CREATE TABLE episode(
191     episode_id INT NOT NULL,
192     season_id INT NOT NULL,
193     img_id INT NOT NULL,
194     PRIMARY KEY (episode_id,season_id),
195     FOREIGN KEY (img_id) REFERENCES images(id) ON DELETE RESTRICT ON UPDATE CASCADE
196 );
197 CREATE TABLE episode_entries(
198     entry_id INT AUTO_INCREMENT PRIMARY KEY,
199     episode_id INT NOT NULL,
200     season_id INT NOT NULL,
201     nt_name VARCHAR(30) NOT NULL,
202     chef_id INT NOT NULL,
203     rc_id INT NOT NULL,
204     score1 INT NOT NULL CHECK(score1 >= 1 AND score1 <=5),
205     score2 INT NOT NULL CHECK(score2 >= 1 AND score2 <=5),
206     score3 INT NOT NULL CHECK(score3 >= 1 AND score3 <=5),
207     total_score INT GENERATED ALWAYS AS (score1 + score2 + score3) STORED,
208     FOREIGN KEY (episode_id,season_id) REFERENCES episode(episode_id,season_id) ON DELETE RESTRICT ON UPDATE CASCADE,
209     FOREIGN KEY (rc_id) REFERENCES recipe(recipe_id) ON DELETE RESTRICT ON UPDATE CASCADE,
210     FOREIGN KEY (chef_id) REFERENCES chef(chef_id) ON DELETE RESTRICT ON UPDATE CASCADE,
211     FOREIGN KEY (nt_name) REFERENCES national_cuisine(nt_name) ON DELETE RESTRICT ON UPDATE CASCADE
212 );
213 CREATE TABLE judges(
214     episode_id INT NOT NULL,
215     season_id INT NOT NULL,
216     first_judge_id INT NOT NULL,
217     second_judge_id INT NOT NULL,

```

```

218 third_judge_id INT NOT NULL,
219 PRIMARY KEY (episode_id,season_id),
220 FOREIGN KEY (episode_id,season_id) REFERENCES episode(episode_id,season_id) ON DELETE RESTRICT ON UPDATE CASCADE,
221 FOREIGN KEY (first_judge_id) REFERENCES chef(chef_id) ON DELETE RESTRICT ON UPDATE CASCADE,
222 FOREIGN KEY (second_judge_id) REFERENCES chef(chef_id) ON DELETE RESTRICT ON UPDATE CASCADE,
223 FOREIGN KEY (third_judge_id) REFERENCES chef(chef_id) ON DELETE RESTRICT ON UPDATE CASCADE
224 );
225
226 /*-----INDEXES-----*/
227 CREATE INDEX idx_age ON chef(age);
228
229 DELIMITER //
230 /*-----PROCEDURES-----*/
231 CREATE PROCEDURE update_recipe_nutritional_info_after_insert( IN rc_id INT, IN ing_id INT, IN amount NUMERIC(10,4), IN unit INT )
232 BEGIN
233     DECLARE calories NUMERIC(8,2);
234     DECLARE fat NUMERIC(8,2);
235     DECLARE carbs NUMERIC(8,2);
236     DECLARE protein NUMERIC(8,2);
237
238     -- Calculate the nutritional values of the ingredient based on its amount and unit
239     SELECT
240         (amount * ing.calories_per_100g * uom.conversion_rate / 100),
241         (amount * ing.fat_per_100g * uom.conversion_rate / 100),
242         (amount * ing.carbs_per_100g * uom.conversion_rate / 100),
243         (amount * ing.protein_per_100g * uom.conversion_rate / 100)
244     INTO
245         calories, fat, carbs, protein
246     FROM
247         ingredient ing
248     JOIN
249         unit_of_measure uom ON uom.u_o_m_id = unit
250     WHERE
251         ing.ing_id = ing_id;
252
253     -- Update the recipe table with the new nutritional values
254     UPDATE recipe
255     SET
256         calories_per_portion = calories_per_portion + calories/portions,
257         fat_per_portion = fat_per_portion + fat/portions,
258         carbs_per_portion = carbs_per_portion + carbs/portions,
259         protein_per_portion = protein_per_portion + protein/portions
260     WHERE
261         recipe_id = rc_id;
262 END;
263 //
264 CREATE PROCEDURE update_recipe_nutritional_info_after_delete( IN rc_id INT, IN ing_id INT, IN amount NUMERIC(10,4), IN unit INT)
265 BEGIN
266     DECLARE calories NUMERIC(8,2);
267     DECLARE fat NUMERIC(8,2);
268     DECLARE carbs NUMERIC(8,2);
269     DECLARE protein NUMERIC(8,2);
270
271     -- Calculate the nutritional values of the ingredient based on its amount and unit
272     SELECT
273         (amount * ing.calories_per_100g * uom.conversion_rate / 100),
274         (amount * ing.fat_per_100g * uom.conversion_rate / 100),
275         (amount * ing.carbs_per_100g * uom.conversion_rate / 100),
276         (amount * ing.protein_per_100g * uom.conversion_rate / 100)
277     INTO
278         calories, fat, carbs, protein
279     FROM
280         ingredient ing
281     JOIN
282         unit_of_measure uom ON uom.u_o_m_id = unit
283     WHERE
284         ing.ing_id = ing_id;
285
286     -- Update the recipe table by subtracting the nutritional values of the deleted ingredient

```



```

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

```

```

356 -- Check if the chef has participated in the last three episodes
357 IF EXISTS (
358     SELECT 1
359     FROM episode_entries
360     WHERE chef_id = NEW.chef_id AND season_id = NEW.season_id AND episode_id IN (new.episode_id, new.episode_id - 1, new.
361         episode_id - 2)
362     GROUP BY chef_id, season_id
363     HAVING COUNT(*) >= 3
364 ) THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Chef cannot compete in more than 3 consecutive episodes in the same
365     season.';
366 END IF;
367
368 -- Check if the recipe has participated in the last three episodes
369 IF EXISTS (
370     SELECT 1
371     FROM episode_entries
372     WHERE rc_id = NEW.rc_id AND season_id = NEW.season_id AND episode_id IN (new.episode_id, new.episode_id - 1, new.
373         episode_id - 2)
374     GROUP BY rc_id, season_id
375     HAVING COUNT(*) >= 3
376 ) THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Recipe cannot compete in more than 3 consecutive episodes in the same
377     season.';
378 END IF;
379
380 -- Check if the national cuisine has participated in the last three episodes
381 IF EXISTS (
382     SELECT 1
383     FROM episode_entries
384     WHERE nt_name = NEW.nt_name AND season_id = NEW.season_id AND episode_id IN (new.episode_id, new.episode_id - 1, new.
385         episode_id - 2)
386     GROUP BY nt_name, season_id
387     HAVING COUNT(*) >= 3
388 ) THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'A national cuisine cannot compete in more than 3 consecutive episodes
389     in the same season.';
390 END IF;
391
392 END//
393 CREATE TRIGGER consecutive_judges_check BEFORE INSERT ON judges
394 FOR EACH ROW
395 BEGIN
396
397     -- Check if the first judge has participated in the last three episodes
398     IF EXISTS (
399         SELECT 1
400         FROM judges
401         WHERE (first_judge_id = NEW.first_judge_id OR second_judge_id = NEW.first_judge_id OR third_judge_id = NEW.
402             first_judge_id ) AND
403             season_id = NEW.season_id AND episode_id IN (new.episode_id, new.episode_id - 1, new.episode_id - 2)
404         GROUP BY season_id
405         HAVING COUNT(*) >= 3) THEN
406         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Judge cannot compete in more than 3 consecutive episodes in the same
407             season.';
408     END IF;
409
410     -- Check if the second judge has participated in the last three episodes
411     IF EXISTS (
412         SELECT 1
413         FROM judges
414         WHERE (first_judge_id = NEW.second_judge_id OR second_judge_id = NEW.second_judge_id OR third_judge_id = NEW.
415             second_judge_id ) AND
416             season_id = NEW.season_id AND episode_id IN (new.episode_id, new.episode_id - 1, new.episode_id - 2)
417         GROUP BY season_id
418         HAVING COUNT(*) >= 3
419 ) THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Judge cannot compete in more than 3 consecutive episodes in the same
420     season.';
421 END IF;
422
423     -- Check if the third judge has participated in the last three episodes
424     IF EXISTS (
425         SELECT 1

```

```

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

```

Listing 1: DDL Script

8. DML script

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

```

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

```

Listing 2: DML Script (for table `ingredient_group`)

Views

Τα Views είναι εικονικοί πίνακες που δημιουργούνται με τη βοήθεια κάποιων query και αποθηκεύονται στη βάση για μετέπειτα χρήση, χωρίς όμως να αποθηκεύουν δεδομένα.

Στη βάση μας εισάγαμε ένα view το οποίο περιέχει όλα τα άτομα που συμμετείχαν είτε ως μάγειρες είτε ως κριτές σε κάθε επεισόδιο. Η δημιουργία του query αυτού βρίσκεται στο DML script,.

```
1 CREATE VIEW chefs_in_episodes (episode_id, season_id, chef_id, job) AS
2     SELECT episode_id, season_id, chef_id, "chef" as job
3     FROM episode_entries
4     UNION ALL
5     SELECT episode_id, season_id, first_judge_id as chef_id, "judge" as job
6     FROM judges
7     UNION ALL
8     SELECT episode_id, season_id, second_judge_id as chef_id, "judge" as job
9     FROM judges
10    UNION ALL
11    SELECT episode_id, season_id, third_judge_id as chef_id, "judge" as job
12    FROM judges;
```

Listing 3: DML Script (Views)

9. Random Episode Generator

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

Οι εν λόγω κώδικες παρατίθενται παρακάτω:

```
1 const mysql = require('mysql');
2
3 // Create MySQL connection
4 const connection = mysql.createConnection({
5   host: 'localhost',
6   user: 'root',
7   password: 'password',
8   database: 'cooking_competition'
9 });
10
11 // Connect to MySQL
12 connection.connect((err) => {
13   if (err) {
14     console.error('Error connecting to database: ' + err.stack);
15     return;
16   }
17
18   console.log('Connected to database');
19
20
21 // Define async function to generate data
22 const generateData = async () => {
23   try {
24     // Define async function to generate data for a single season
25     const generateDataForSeason = async (season, nationalCuisines) => {
26       // Fetch participation in previous episodes
```

```

27     const previousEpisodesChef = {};
28     const previousEpisodesNt = {};
29     const previousEpisodesRc = {};
30     // Function to generate data for a single episode
31     const generateDataForEpisode = async (episode, previousEpisodesChef, previousEpisodesRc, previousEpisodesNt) => {
32         console.log(`Generating data for Season ${season}, Episode ${episode}`);
33
34         const episodeCuisines = [];
35
36         // List to keep track of selected chefs for this episode
37         const selectedChefs = [];
38
39         // Loop until we have selected 10 unique cuisines for this episode
40         while (episodeCuisines.length < 10) {
41             // Find a valid national cuisine not participating in the last 3 episodes
42             const validCuisine = await findValidCuisine(previousEpisodesNt, episode, nationalCuisines);
43
44             // Check if the cuisine is not already selected for this episode
45             if (!episodeCuisines.includes(validCuisine.nt_name)) {
46                 episodeCuisines.push(validCuisine.nt_name);
47
48                 let chef_id;
49                 let recipe;
50
51                 // Attempt to find a valid chef and recipe until successful or exhausted
52                 let attempts = 0;
53                 while (attempts < 25) {
54                     // Find a valid chef for the cuisine not participating in the last 3 episodes
55                     chef_id = await findValidChef(validCuisine.nt_name, previousEpisodesChef, episode, selectedChefs);
56                     ;
57
58                     // Find a valid recipe for the cuisine and chef not participating in the last 3 episodes
59                     recipe = await findValidRecipe(validCuisine.nt_name, chef_id, previousEpisodesRc, episode);
60
61                     // If a valid recipe is found, break the loop
62                     if (recipe) {
63                         break;
64                     }
65
66                     // Increment attempts and try again with another chef
67                     attempts++;
68                 }
69
70                 // Output episode data
71                 console.log(`Season ${season}, Episode ${episode}, Entry ${episodeCuisines.length}`);
72                 console.log(`National Cuisine: ${validCuisine.nt_name}`);
73                 console.log(`Recipe: ${recipe ? recipe.recipe_name : 'No valid recipe found'}`);
74                 console.log(`Chef: ${chef_id}`);
75                 console.log('-----');
76
77                 // Insert episode data into episode_entries table
78                 const episodeEntry = {
79                     entry_id: (episode - 1) * 10 + (season - 1) * 100 + episodeCuisines.length,
80                     episode_id: episode,
81                     season_id: season,
82                     nt_name: validCuisine.nt_name,
83                     chef_id: chef_id,
84                     rc_id: recipe ? recipe.recipe_id : null,
85                     score1: generateRandomScore(),
86                     score2: generateRandomScore(),
87                     score3: generateRandomScore()
88                 };
89
90                 await new Promise((resolve, reject) => {
91                     connection.query('INSERT INTO episode_entries SET ?', episodeEntry, (err, result) => {
92                         if (err) reject(err);
93                         console.log(`Inserted episode entry with ID ${result.insertId}`);
94                         resolve();
95                     });
96                 });
97             }
98         }
99     };

```

```

94         });
95     });
96
97     // Add selected chef to the list
98     selectedChefs.push(chef_id);
99     if (!previousEpisodesChef[chef_id]) {
100         previousEpisodesChef[chef_id] = [];
101     }
102     if (!previousEpisodesRc[recipe.recipe_id]) {
103         previousEpisodesRc[recipe.recipe_id] = [];
104     }
105     if (!previousEpisodesNt[nationalCuisines.indexOf(validCuisine.nt_name)]) {
106         previousEpisodesNt[nationalCuisines.indexOf(validCuisine.nt_name)] = [];
107     }
108     previousEpisodesChef[chef_id].push(episode);
109     previousEpisodesRc[recipe.recipe_id].push(episode);
110     previousEpisodesNt[nationalCuisines.indexOf(validCuisine.nt_name)].push(episode);
111
112     }
113 }
114 };
115
116
117
118 // Generate data for each episode of the current season
119 for (let episode = 1; episode <= 10; episode++) {
120     await generateDataForEpisode(episode, previousEpisodesChef, previousEpisodesRc, previousEpisodesNt);
121 }
122 };
123
124 // Function to find a valid national cuisine not participating in the last 3 episodes
125 const findValidCuisine = async (previousEpisodes, episode, nationalCuisines) => {
126     let cuisine;
127     let selectedCuisine = false;
128
129     while (!selectedCuisine) {
130         cuisine = await new Promise((resolve, reject) => {
131             connection.query('SELECT * FROM national_cuisine ORDER BY RAND() LIMIT 1', (err, result) => {
132                 if (err) reject(err);
133                 resolve(result[0]);
134             });
135         });
136
137         // Check if the cuisine has participated in the last 3 episodes
138         const cuisineName = cuisine.nt_name;
139         if (!previousEpisodes[nationalCuisines.indexOf(cuisineName)] || !previousEpisodes[nationalCuisines.indexOf(cuisineName)].includes(episode - 1) || !previousEpisodes[nationalCuisines.indexOf(cuisineName)].includes(episode - 2) || !previousEpisodes[nationalCuisines.indexOf(cuisineName)].includes(episode - 3)) {
140             selectedCuisine = true;
141         }
142     }
143
144     return cuisine;
145 };
146
147 const findValidChef = async (cuisineName, previousEpisodes, episode, selectedChefs) => {
148     let chef_id;
149     let selectedChef = false;
150
151     while (!selectedChef) {
152         // Select a random chef for the cuisine
153         const chefResult = await new Promise((resolve, reject) => {
154             connection.query('SELECT * FROM chef_national_cuisines INNER JOIN chef ON chef_national_cuisines.chef_id = chef.chef_id WHERE nt_name = ? ORDER BY RAND() LIMIT 1', [cuisineName], (err, result) => {
155                 if (err) reject(err);
156                 resolve(result);
157             });
158         });

```

```

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

```

```

223 generateData();
224 });

```

Listing 4: random.js Script

```

1  const mysql = require('mysql');
2
3  // Create MySQL connection
4  const connection = mysql.createConnection({
5    host: 'localhost',
6    user: 'root',
7    password: 'password',
8    database: 'cooking_competition'
9  });
10
11 // Connect to MySQL
12 connection.connect((err) => {
13   if (err) {
14     console.error('Error connecting to database: ' + err.stack);
15     return;
16   }
17   console.log('Connected to database');
18
19   // Function to shuffle an array
20   const shuffle = (array) => {
21     for (let i = array.length - 1; i > 0; i--) {
22       const j = Math.floor(Math.random() * (i + 1));
23       [array[i], array[j]] = [array[j], array[i]];
24     }
25     return array;
26   };
27
28   // Define async function to generate judges for each episode
29   const generateJudgesForEpisode = async (episode, season, previousJudges) => {
30     console.log(`Generating judges for Season ${season}, Episode ${episode}`);
31
32     // Retrieve random chefs who are eligible to be judges
33     const eligibleJudges = await new Promise((resolve, reject) => {
34       connection.query('SELECT * FROM chef WHERE NOT EXISTS (SELECT * FROM episode_entries WHERE episode_id = ? AND season_id = ? AND chef_id = chef.chef_id) ORDER BY RAND() LIMIT 3', [episode, season], (err, result) => {
35         if (err) reject(err);
36         resolve(result);
37       });
38     });
39
40     // Shuffle the eligible judges randomly
41     const shuffledJudges = shuffle(eligibleJudges);
42
43     // Select the first three judges
44     const judges = shuffledJudges.slice(0, 3);
45
46     // Output judges data
47     console.log(`Judges for Season ${season}, Episode ${episode}:`);
48     console.log(`First Judge: ${judges[0].chef_id}`);
49     console.log(`Second Judge: ${judges[1].chef_id}`);
50     console.log(`Third Judge: ${judges[2].chef_id}`);
51     console.log('-----');
52
53     // Insert judges data into the judges table
54     const judgesEntry = {
55       episode_id: episode,
56       season_id: season,
57       first_judge_id: judges[0].chef_id,
58       second_judge_id: judges[1].chef_id,
59       third_judge_id: judges[2].chef_id
60     };
61
62     await new Promise((resolve, reject) => {

```



```

63     connection.query('INSERT INTO judges SET ?', judgesEntry, (err, result) => {
64         if (err) reject(err);
65         console.log('Inserted judges entry for Episode ${episode} of Season ${season}');
66         resolve();
67     });
68 });
69
70
71 // Define async function to generate judges for each season
72 const generateJudgesForSeason = async (season) => {
73     // Fetch participation of judges in previous episodes
74     const previousJudges = {};
75     const judges = await new Promise((resolve, reject) => {
76         connection.query('SELECT * FROM judges WHERE season_id = ?', [season], (err, result) => {
77             if (err) reject(err);
78             resolve(result);
79         });
80     });
81
82     judges.forEach((judge) => {
83         const episode_id = judge.episode_id;
84         if (!previousJudges[episode_id]) {
85             previousJudges[episode_id] = [];
86         }
87         previousJudges[episode_id].push(judge.first_judge_id);
88         previousJudges[episode_id].push(judge.second_judge_id);
89         previousJudges[episode_id].push(judge.third_judge_id);
90     });
91
92     // Generate judges for each episode of the current season
93     for (let episode = 1; episode <= 10; episode++) {
94         await generateJudgesForEpisode(episode, season, previousJudges);
95     }
96 });
97
98 // Define async function to generate judges for all seasons
99 const generateJudges = async () => {
100     for (let season = 1; season <= 5; season++) {
101         console.log('Generating judges for Season ${season}');
102         await generateJudgesForSeason(season);
103     }
104 };
105
106 // Call the async function to generate judges
107 generateJudges();
108 });

```

Listing 5: random_judges.js Script

10. Permissions

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

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

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

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

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

Παρατίθεται παρακάτω ο κώδικας από το συγκεκριμένο αρχείο:

```

1 DELIMITER //
2 -- Procedure to update chef information
3 CREATE PROCEDURE update_chef_info(IN id INT, IN new_name VARCHAR(50), IN new_surname VARCHAR(50), IN new_date_of_birth date, IN
4   new_age INT, IN new_years_of_expertice INT, IN new_professional_title VARCHAR(20), IN new_img_id INT)
5 BEGIN
6   IF (id = CURRENT_USER_ID()) THEN
7     UPDATE chef SET name = new_name, surname = new_surname, _date_of_birth = new_date_of_birth, age = new_age,
8       years_of_expertice = new_years_of_expertice, professional_title = new_professional_title, img_id = new_img_id WHERE id =
9       chef_id;
10  ELSE
11    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Permission denied';
12  END IF;
13 END //
14 -- Procedure to update a recipe
15 CREATE PROCEDURE update_recipe(IN recip_id INT, IN c_id INT, IN new_name VARCHAR(100), IN new_description VARCHAR(500), IN
16   new_recipe_type VARCHAR(20), IN new_difficulty INT, IN new_ INT,
17   IN new_exec_time INT, IN new_basic_ingredient INT, IN new_nt VARCHAR(30), IN new_portions INT, IN new_img_id INT)
18 BEGIN
19   IF ( (EXISTS(SELECT * FROM chef_recipe WHERE (chef_id = c_id AND rc_id = recip_id))) AND c_id = CURRENT_USER_ID()) THEN
20     UPDATE recipe SET name = new_name, description = new_description, recipe_type = new_recipe_type, difficulty =
21       new_difficulty, prep_time = new_prep_time,
22       exec_time = new_exec_time, basic_ingredient = new_basic_ingredient, national_cuisine = new_nt, portions = new_portions,
23       img_id = new_img_id WHERE recipe_id = recip_id;
24   ELSE
25     SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Permission denied';
26   END IF;
27 END //
28 DELIMITER ;
29 CREATE ROLE chef_role;
30 GRANT EXECUTE ON PROCEDURE update_chef_info TO chef_role;
31 GRANT EXECUTE ON PROCEDURE update_recipe TO chef_role;
32 GRANT INSERT ON cooking_competition.recipe TO chef_role;
33 GRANT INSERT ON cooking_competition.ingredients_used TO chef_role;

```

```
31 GRANT INSERT ON cooking_competition.equipment_used TO chef_role;
32 GRANT INSERT ON cooking_competition.recipe_label TO chef_role;
33 GRANT INSERT ON cooking_competition.recipe_meal_type TO chef_role;
34 GRANT INSERT ON cooking_competition.recipe_thematic_section TO chef_role;
35 GRANT INSERT ON cooking_competition.recipe_tips TO chef_role;
36 GRANT INSERT ON cooking_competition.step TO chef_role;
37 DELIMITER //
38 CREATE PROCEDURE create_chef_users()
39 BEGIN
40     DECLARE done INT DEFAULT FALSE;
41     DECLARE chef_id INT;
42     DECLARE chef_username VARCHAR(255);
43     DECLARE chef_password VARCHAR(255);
44     DECLARE cur CURSOR FOR SELECT c.chef_id, c.username, u.password FROM chef c INNER JOIN users u ON c.username = u.username;
45     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
46
47     OPEN cur;
48
49     read_loop: LOOP
50         FETCH cur INTO chef_id, chef_username, chef_password;
51         IF done THEN
52             LEAVE read_loop;
53         END IF;
54
55         -- Dynamic SQL to create user
56         SET @sql = CONCAT('CREATE USER ', chef_username, '@localhost IDENTIFIED BY ', chef_password, ';');
57         PREPARE stmt FROM @sql;
58         EXECUTE stmt;
59         DEALLOCATE PREPARE stmt;
60
61         -- Grant necessary permissions to the user
62         SET @grant_sql = CONCAT('GRANT chef_role TO ', chef_username, '@localhost;');
63         PREPARE grant_stmt FROM @grant_sql;
64         EXECUTE grant_stmt;
65         DEALLOCATE PREPARE grant_stmt;
66     END LOOP;
67
68     CLOSE cur;
69 END //
70 DELIMITER ;
71 CREATE USER 'admin_1'@'localhost' IDENTIFIED BY 'W3lc0m3';
72 CREATE USER 'admin_2'@'localhost' IDENTIFIED BY 'I4mth3b3st';
73 GRANT ALL PRIVILEGES ON cooking_competition.* TO 'admin_1'@'localhost';
74 GRANT ALL PRIVILEGES ON cooking_competition.* TO 'admin_2'@'localhost';
75 CALL create_chef_users;
```

Listing 6: roles.sql

11. Queries-Ερωτήματα

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

11.1

Εκφώνηση:

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

Κώδικας:

```

1 SELECT chef.chef_id,name,surname, avg(total_score/3) average_score
2 FROM episode_entries
3 INNER JOIN chef
4 ON episode_entries.chef_id = chef.chef_id GROUP BY chef.chef_id;
5
6 SELECT nt_name, avg(total_score/3) average_score
7 FROM episode_entries
8 GROUP BY nt_name;

```

Listing 7: Query 1

Αποτελέσματα:

chef_id	name	surname	average_score
1	Lefteris	Lazareu	2.77777778
2	Viannis	Lucacos	2.91666667
3	Dimitris	Skarwoutos	3.16666667
4	Sotiris	Kontizas	3.83333333
5	Panos	Ioannidis	3.89898989
6	Leonidas	Koutsopoulos	2.96969697
7	Margarita	Nikolaidi	2.95833333
8	Giovanni	Scaraggi	4.00000000
9	Akis	Petretzidis	2.83333333
10	Christos	Glossidis	3.16666667
11	Giorgos	Porfiris	2.77777778
12	Stavros	Varthalitis	2.83333333
13	Stavros	Georgiou	3.87407407
14	Spiridoula	Karampoutaki	3.80000000
15	Ilias	Maziolis	2.94444444
16	Maria	Day	3.22222222
17	Ioannis	Charalampous	3.86666667
18	Dimitris	Bellos	2.38895238
19	John	Reano	2.89393939
20	Maria	Lazaridou	2.85185185
21	Panagiotis	Tzamalīs	2.73333333
22	Pavlos	Happilos	2.77777778
23	Stergios	Nilas	3.44444444
24	Grigorios	Giannopoulos	3.48748741
25	Nikos	Trakas	2.77777778
26	Panagiotis	Avgenikos	3.28571429
27	Vaggelis	Filaktakis	2.66666667
28	Dimitris	Politakis	2.68000000
29	Giannis	Chatzias	2.83333333
30	Nikitas	Filippakis	3.19047619

30	Nikitas	Filippakis	3.19047619
31	Konstantinos	Kokkotas	2.83333333
32	Alexandros	Antonoulakis	3.12626013
33	Alkistis	Alexaki	2.86111111
34	Asimilina	Oustalli	2.97222222
35	Giorgos	Orfanos	3.86666667
36	Zahir	Vavary	3.76198476
37	Manos	Sarris	2.42424242
38	Panos	Togias	2.58888889
39	Padelis	Vouros	3.88888888
40	Stamatis	Kovalos	3.28888888
41	Tasos	Kiriakakis	3.39393939
42	Charalambos	Kotsonis	3.88888888
43	Xristos	Barbas	3.22222222
44	Giorgos	Chronakis	3.33333333
45	Gogo	Delogianni	3.12500000
46	Dimitris	Papapanagiotou	3.16666667
47	Zachos	Papadakis	2.75757576
48	Lambros	Vakisios	3.15151515
49	Levis	Kodhima	3.14285714
50	Ektoras	Botrini	3.18888889
51	Adas	Kontovas	3.16666667
52	Athinagoras	Kostakos	2.98476190
53	Aris	Vezenes	3.68888888
54	Nikos	Thomas	3.24242424
55	Christina	Christophi	3.88333333
56	Ilias	Panagiotou	2.88888889
57	Lefteris	Zafeiropoulos	3.18518518
58	Marlos	Ioannou	2.85185185
59	Christos	Moiras	3.18518518
60	Kostas	Fretzos	3.86666667

nt_name	average_score
American	2.79166667
Arab	3.25396825
Australian	2.97101449
Brazilian	3.01234568
British	2.77777778
Chinese	3.14583333
Cypriot	2.90476190
Egyptian	3.06349206
French	2.83870968
German	2.92424242
Greek	2.94666667
Indian	2.95833333
Italian	3.06944444
Japanese	3.04938272
Korean	3.32898765
Lebanese	3.04545455
Mexican	3.15942029
Polish	3.01234568
Russian	3.10256410
South African	2.98245614
Spanish	3.02564103

11.2**Εκφώνηση:**

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

Παραδοχές:

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

Κώδικας:

```

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

```

Listing 8: Query 2

Αποτελέσματα:

```
mysql> CALL GetChefsByCuisineAndSeason("Chinese",2);
```

chef_id	name	surname
24	Grhgorhs	Giannopoulos
44	Giorgos	Chronakis
4	Sotiris	Kontizas
40	Stamatis	Kovaiois
2	Yiannis	Lucacos
22	Pavlos	Happilos
60	Kostas	Fretzios

7 rows in set (0.01 sec)

```
mysql> CALL GetChefsByCuisineAndSeason("Chinese",4);
```

chef_id	name	surname
24	Grhgorhs	Giannopoulos
44	Giorgos	Chronakis
4	Sotiris	Kontizas
40	Stamatis	Kovaiois
2	Yiannis	Lucacos
22	Pavlos	Happilos
60	Kostas	Fretzios

7 rows in set (0.00 sec)

```
mysql> CALL GetChefsByCuisineAndSeason("Greek",5);
```

chef_id	name	surname
9	Akis	Petretzikis
2	Yiannis	Lucacos
21	Panagiotis	Tzamalīs
22	Pavlos	Happilos
1	Lefteris	Lazarou
29	Giannis	Chatzikas
49	Levis	Kodhima
42	Charalambos	Kotsonis

8 rows in set (0.00 sec)

```
mysql> CALL GetChefsByCuisineAndSeason("Chinese",2);
```

chef_id	name	surname
2	Yiannis	Lucacos
4	Sotiris	Kontizas
40	Stamatis	Kovaiois

3 rows in set (0.02 sec)

```
mysql> CALL GetChefsByCuisineAndSeason("Chinese",4);
```

chef_id	name	surname
40	Stamatis	Kovaiois
44	Giorgos	Chronakis
60	Kostas	Fretzios

3 rows in set (0.01 sec)

```
mysql> CALL GetChefsByCuisineAndSeason("Greek",5);
```

chef_id	name	surname
1	Lefteris	Lazarou
2	Yiannis	Lucacos
29	Giannis	Chatzikas

3 rows in set (0.02 sec)

11.3

Εκφώνηση:

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

Κώδικας:

```

1 WITH num_rec AS (
2     SELECT MAX(count) as max_count
3     FROM (
4         SELECT COUNT(rc_id) as count
5         FROM chef_recipe
6         INNER JOIN chef ON chef.chef_id = chef_recipe.chef_id
7         WHERE chef.age < 30
8         GROUP BY chef.chef_id
9     ) as rec_per_chef
10 )
11 SELECT chef.chef_id, chef.name, chef.surname, COUNT(rc_id) as count
12 FROM chef_recipe
13 INNER JOIN chef ON chef.chef_id = chef_recipe.chef_id
14 WHERE chef.age < 30
15 GROUP BY chef.chef_id, chef.name, chef.surname
16 HAVING COUNT(rc_id) = (SELECT max_count FROM num_rec);

```

Listing 9: Query 3

Αποτελέσματα:

chef_id	name	surname	count
29	Giannis	Chatzikas	4
24	Grhgorhs	Giannopoulos	4

11.4

Εκφώνηση:

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

Κώδικας:

```

1 SELECT chef_id, name , surname
2 FROM chef
3 WHERE chef_id NOT IN(
4 SELECT first_judge_id FROM judges
5 UNION
6 SELECT second_judge_id FROM judges
7 UNION
8 SELECT third_judge_id FROM judges
9 );

```

Listing 10: Query 4

Αποτελέσματα:

chef_id	name	surname
24	Grhgorhs	Giannopoulos
39	Padelis	Vouros
60	Kostas	Fretzios

3 rows in set (0.01 sec)

11.5

Εκφώνηση:

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

Κώδικας:

```

1 WITH appearances AS (
2 SELECT season_id,judge_id, COUNT(episode_id) AS appearances_of_judge
3 FROM (
4 SELECT episode_id,season_id, first_judge_id as judge_id
5 FROM judges
6 UNION ALL
7 SELECT episode_id,season_id, second_judge_id as judge_id
8 FROM judges
9 UNION ALL
10 SELECT episode_id,season_id, third_judge_id as judge_id
11 FROM judges) AS judges_combined
12 GROUP BY season_id,judge_id
13 ),
14 judge_appearances AS (

```

```

15      SELECT a.season_id as season_id, a.judge_id as judge_id , c.name as name , c.surname as surname, a.appearances_of_judge
      as appearances_of_judges
16      FROM appearances a
17      INNER JOIN chef c ON a.judge_id = c.chef_id
18      WHERE a.appearances_of_judge > 3
19      GROUP BY season_id, judge_id)
20 SELECT j1.season_id, j1.judge_id as judge1_id, j1.name as judge1_name, j1.surname as judge1_surname, j2.judge_id as judge2_id, j2.
      name as judge2_name, j2.surname as judge2_surname, j1.appearances_of_judges
21 FROM judge_appearances j1
22 JOIN judge_appearances j2
23 ON j1.season_id = j2.season_id AND j1.judge_id < j2.judge_id
24 WHERE j1.appearances_of_judges = j2.appearances_of_judges ;

```

Listing 11: Query 5

Αποτελέσματα:

season_id	judge1_id	judge1_name	judge1_surname	judge2_id	judge2_name	judge2_surname	appearances_of_judges
2	8	Giovanni	Scaraggi	12	Stavros	Varthalitis	4

1 row in set (0.00 sec)

11.6

Εκφώνηση:

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

Παραδοχές:

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

Κώδικας:

```

1 SELECT a.lb_name AS label1,
2        b.lb_name AS label2,
3        COUNT(*) AS count
4 FROM episode_entries e
5 INNER JOIN recipe_label a ON e.rc_id = a.rc_id
6 INNER JOIN recipe_label b ON (a.lb_name < b.lb_name AND a.rc_id = b.rc_id)
7 GROUP BY label1, label2
8 ORDER BY count DESC
9 LIMIT 3;
10
11 /*Alternate Query Plan*/
12 SELECT a.lb_name AS label1,

```

```

13      b.lb_name AS label2,
14      COUNT(*) AS count
15 FROM episode_entries e IGNORE INDEX(rc_id)
16 INNER JOIN recipe_label a IGNORE INDEX(PRIMARY,lb_name) ON e.rc_id = a.rc_id
17 INNER JOIN recipe_label b IGNORE INDEX(PRIMARY,lb_name) ON (a.lb_name < b.lb_name AND a.rc_id = b.rc_id)
18 GROUP BY label1, label2
19 ORDER BY count DESC
20 LIMIT 3;

```

Listing 12: Query 6

Αποτελέσματα:

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

label1	label2	count
Egg Free Diet	Nuts Free Diet	122
Egg Free Diet	Main Course	79
Nuts Free Diet	Vegetarian Diet	71

3 rows in set (0.01 sec)

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

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	a	NULL	index	PRIMARY,lb_name	lb_name	122	NULL	162	100.00	Using index; Using temporary; Using filesort
1	SIMPLE	b	NULL	ref	PRIMARY,lb_name	PRIMARY	4	cooking_competition.a.rc_id	2	33.33	Using where; Using index
1	SIMPLE	e	NULL	ref	rc_id	rc_id	4	cooking_competition.a.rc_id	7	100.00	Using index
3 rows in set, 1 warning (0.00 sec)											
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	a	NULL	ALL	NULL	NULL	NULL	NULL	162	100.00	Using temporary; Using filesort
1	SIMPLE	b	NULL	ALL	NULL	NULL	NULL	NULL	162	0.62	Using where; Using join buffer (hash join)
1	SIMPLE	e	NULL	ALL	NULL	NULL	NULL	NULL	500	1.49	Using where; Using join buffer (hash join)
3 rows in set, 1 warning (0.00 sec)											

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

11.7

Εκφώνηση:

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

Κώδικας:


```

1 WITH max_appearances AS(
2     SELECT MAX(count_appearances) as max_count
3     FROM(
4         SELECT COUNT(chef_id) as count_appearances
5         FROM episode_entries
6         GROUP BY chef_id) AS chef_appearances
7 )
8 SELECT c.chef_id, c.name, c.surname, COUNT(e.chef_id) as appearances
9 FROM episode_entries e
10 INNER JOIN chef c ON e.chef_id = c.chef_id
11 GROUP BY chef_id
12 HAVING appearances < ((SELECT max_count FROM max_appearances) - 4);

```

Listing 13: Query 7

Αποτελέσματα:

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

49 rows in set (0.01 sec)

11.8

Εκφώνηση:

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

Παραδοχές:

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

Κώδικας:

```

1 SELECT ep.episode_id, ep.season_id , COUNT(*) as number_of_eq
2 FROM episode_entries ep
3 INNER JOIN equipment_used eq
4 ON ep.rc_id = eq.rc_id
5 GROUP BY ep.episode_id,ep.season_id
6 ORDER BY number_of_eq DESC
7 LIMIT 1;
8
9 /*Alternate Query Plan*/
10 SELECT ep.episode_id, ep.season_id , COUNT(*) as number_of_eq
11 FROM episode_entries ep IGNORE INDEX(rc_id,episode_id)
12 INNER JOIN equipment_used eq IGNORE INDEX(PRIMARY,eq_id)
13 ON ep.rc_id = eq.rc_id
14 GROUP BY ep.episode_id,ep.season_id
15 ORDER BY number_of_eq DESC
16 LIMIT 1;

```

Listing 14: Query 8

Αποτελέσματα:

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

episode_id	season_id	number_of_eq
10	5	39

1 row in set (0.01 sec)

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

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	ep	NULL	ALL	episode_id,rc_id	NULL	NULL	NULL	500	100.00	Using temporary; Using filesort
1	SIMPLE	eq	NULL	ref	PRIMARY	PRIMARY	4	cooking_competition.ep.rc_id	2	100.00	Using index
2 rows in set, 1 warning (0.00 sec)											
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	eq	NULL	ALL	NULL	NULL	NULL	NULL	190	100.00	Using temporary; Using filesort
1	SIMPLE	ep	NULL	ALL	NULL	NULL	NULL	NULL	500	1.49	Using where; Using join buffer (hash join)
2 rows in set, 1 warning (0.00 sec)											

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

11.9**Εκφώνηση:**

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

Παραδοχές:

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

Κώδικας:

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

Listing 15: Query 9

Αποτελέσματα:

season_id	average_carbs_per_season
1	20.804800
2	21.774400
3	24.084800
4	22.460300
5	24.573500

5 rows in set (0.01 sec)

11.10**Εκφώνηση:**

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

Κώδικας:

```
1 WITH nt_appearances AS (
2     SELECT nt_name, season_id, COUNT(episode_id) AS nt_app_per_season
3     FROM episode_entries
4     GROUP BY nt_name, season_id
5     HAVING COUNT(episode_id) >= 3
6 ),
7 consecutive_appearances AS (
8     SELECT n1.nt_name, n1.nt_app_per_season + n2.nt_app_per_season AS consecutive_appearances, n1.season_id AS season_1, n2.
9         season_id AS season_2
10    FROM nt_appearances n1
11   JOIN nt_appearances n2
12   ON n1.nt_name = n2.nt_name AND n1.season_id = n2.season_id - 1
13 ),
14 appearance_combinations AS (
15     SELECT
16         consecutive_appearances, season_1, COUNT(*) AS combination_count
```

```

16 FROM consecutive_appearances
17 GROUP BY consecutive_appearances, season_1
18 HAVING COUNT(*) > 1
19 )
20 SELECT ca.nt_name, ca.consecutive_appearances, ca.season_1, ca.season_2
21 FROM consecutive_appearances ca
22 JOIN appearance_combinations ac
23 ON ca.consecutive_appearances = ac.consecutive_appearances AND ca.season_1 = ac.season_1
24 ORDER BY ca.season_1, ca.season_2, ca.consecutive_appearances;

```

Listing 16: Query 10

Αποτελέσματα:

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

63 rows in set (0.01 sec)

11.11

Εκφώνηση:

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

Κώδικας:

```

1 WITH judge_score AS(
2   (SELECT e.chef_id as chef_id, e.score1 as score, j.first_judge_id as judge_id
3   FROM episode_entries e
4   INNER JOIN judges j ON (e.episode_id = j.episode_id AND e.season_id = j.season_id))
5   UNION ALL
6   (SELECT e.chef_id as chef_id, e.score2 as score, j.second_judge_id as judge_id
7   FROM episode_entries e
8   INNER JOIN judges j ON (e.episode_id = j.episode_id AND e.season_id = j.season_id))

```

```

9      UNION ALL
10     (SELECT e.chef_id as chef_id, e.score3 as score, j.third_judge_id as judge_id
11      FROM episode_entries e
12      INNER JOIN judges j ON (e.episode_id = j.episode_id AND e.season_id = j.season_id))
13 )
14 SELECT s.chef_id, c.name, c.surname, SUM(s.score) as total_score, s.judge_id, j.name, j.surname
15 FROM judge_score s
16 INNER JOIN chef c ON s.chef_id = c.chef_id
17 INNER JOIN chef j ON s.judge_id = j.chef_id
18 GROUP BY s.chef_id, s.judge_id
19 ORDER BY total_score DESC
20 LIMIT 5;

```

Listing 17: Query 11

Αποτελέσματα:

chef_id	name	surname	total_score	judge_id	name	surname
29	Giannis	Chatzikas	18	8	Giovanni	Scaraggi
12	Stavros	Varthalitis	15	14	Spiridoula	Karampoutaki
48	Lambros	Vakiaros	14	8	Giovanni	Scaraggi
41	Tasos	Kiriklakis	14	21	Panagiotis	Tzamalīs
26	Panagiotis	Avgenikos	14	27	Vaggelis	Filaktakis

5 rows in set (0.04 sec)

11.12

Εκφώνηση:

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

Κώδικας:

```

1  WITH RankedEpisodes AS (
2      SELECT episode_entries.season_id, episode_entries.episode_id, SUM(recipe.difficulty) AS total_difficulty,
3      ROW_NUMBER() OVER (PARTITION BY episode_entries.season_id ORDER BY SUM(recipe.difficulty) DESC) AS row_num
4      FROM
5      episode_entries INNER JOIN recipe
6      ON episode_entries.rc_id=recipe.recipe_id
7      GROUP BY
8      episode_entries.season_id, episode_entries.episode_id
9  )
10 SELECT season_id, episode_id, total_difficulty
11 FROM RankedEpisodes
12 WHERE row_num=1
13 ORDER BY season_id;

```

Listing 18: Query 12

Αποτελέσματα:

season_id	episode_id	total_difficulty
1	6	31
2	8	32
3	7	28
4	2	30
5	10	29

5 rows in set (0.01 sec)

11.13

Εκφώνηση:

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

Κώδικας:

```

1 SELECT e.episode_id, e.season_id,
2       SUM(CASE
3         WHEN c.professional_title = 'chef' THEN 1
4         WHEN c.professional_title = 'sous chef' THEN 2
5         WHEN c.professional_title = 'A chef' THEN 3
6         WHEN c.professional_title = 'B chef' THEN 4
7         WHEN c.professional_title = 'C chef' THEN 5
8         ELSE 0
9       END) AS total_prof_score
10 FROM chefs_in_episodes e
11 INNER JOIN
12 chef c ON e.chef_id=c.chef_id
13 GROUP BY e.episode_id,e.season_id
14 ORDER BY total_prof_score DESC
15 LIMIT 1;

```

Listing 19: Query 13

Αποτελέσματα:

episode_id	season_id	total_prof_score
5	2	47

1 row in set (0.00 sec)

11.14

Εκφώνηση:

Ποια θεματική ενότητα έχει εμφανιστεί τις περισσότερες φορές στο διαγωνισμό;

Κώδικας:

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

Listing 20: Query 14

Αποτελέσματα:

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

11.15**Εκφώνηση:**

Ποιες ομάδες τροφίμων δεν έχουν εμφανιστεί ποτέ στον διαγωνισμό;

Παραδοχές:

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

Κώδικας:

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

Listing 21: Query 15

Αποτελέσματα:

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


12. Οδηγίες Εγκατάστασης Βάσης

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