ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής

Εργασία Μαθήματος ***ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ***

Όνομα φοιτητή	ΑΛΕΞΙΟΣ ΒΑΣΙΛΕΙΟΥ, ΚΩΣΤΑΝΤΙΝΟΣ ΚΟΥΚΟΣ, ΚΡΟΪΤΟΡ ΚΑΤΑΡΤΖΙΟΥ ΙΩΑΝ
Αρ. Μητρώου	Π21009, Π21070, Π21077
Ημερομηνία παράδοσης	29/6/2023

Εκφώνηση εργασίας

1. Εισαγωγή

Έστω η ΒΔ ενός τοπικού ερασιτεχνικού ποδοσφαιρικού συλλόγου στην οποία διατηρούνται πληροφορίες συμμετοχής των μελών του σε διάφορες αθλητικές εκδηλώσεις. Οι πληροφορίες αφορούν τους παίκτες, τους προπονητές, τις ομάδες, τους αγώνες-πρόγραμμα αγώνων κλπ. Πιο συγκεκριμένα: •

• Παίκτες: Για τους παίκτες διατηρούνται πληροφορίες οι οποίες αφορούν το όνομα, επώνυμο, την ομάδα στην οποία ανήκουν, τη θέση στην οποία παίζουν (center back, goal keeper, defender, Center Back, Sweeper/Libero, Right Back, Left Back, κλπ.). Επιπλέον για κάθε παίκτη διατηρούνται συνολικά στατιστικά του με τις κάρτες που έχει λάβει (κίτρινες, κόκκινες κλπ.) καθώς και το συνολικό αριθμό από γκολ που έχει βάλει, συνολικά λεπτά που ήταν ενεργός στον αγώνα κλπ. Το όνομα και το επώνυμο μπορούν να λάβουν μόνον έως 10 χαρακτήρες ελληνικά με πλήρη στίξη (τόνους, διαλυτικά, κλπ.). Δεν θα πρέπει να περιλαμβάνονται περισσότεροι από 11 παίκτες σε κάθε ομάδα. Υπάρχουν και μεταγραφές, κατά συνέπεια ένας παίκτης δεν ανήκει για όλα τα χρόνια στην ίδια ομάδα.

• Προπονητές: Προπονητές στο σύλλογο μπορούν να γίνουν μόνον παλιοί παίκτες του συλλόγου. Οπότε για τους προπονητές διατηρούνται όλες οι πληροφορίες όπως και για τους παίκτες επιπλέον της προπονητικής τους ιδιότητας στην όποια ομάδα του συλλόγου.

• Ομάδες: Για τις ομάδες διατηρούνται πληροφορίες οι οποίες αφορούν το όνομα τους, το γήπεδο της έδρας της, κάποια περιγραφή της ιστορίας τους, καθώς και διάφορα στατιστικά όπως: νίκες εντός/εκτός έδρας, ήττες εντός/εκτός έδρας, ισοπαλίες εντός/εκτός έδρας.

• Αγώνες/πρόγραμμα αγώνων: Για κάθε αγώνα διατηρούνται πληροφορίες όπως ποια είναι η γηπεδούχος και ποια η φιλοξενούμενη ομάδα, ποιο το σκορ της κάθε ομάδας, ποια η ημερομηνία που έγινε ο αγώνας. Επιπλέον θα πρέπει να γίνεται έλεγχος ώστε να μην προγραμματίζονται αγώνες με τις ίδιες ομάδες την ίδια μέρα. Για κάθε ομάδα θα πρέπει να υπάρχει διάστημα 10 ημερών μεταξύ των αγώνων της. Για κάθε αγώνα και για κάθε παίκτη διατηρούνται πληροφορίες όπως τα γκολ που μπήκαν, τα γκολ που ακυρώθηκαν, οι κάρτες (κόκκινες και κίτρινες) που δέχτηκε ένας παίκτης, τα πέναλτι, τα κόρνερ (και σε όλα αυτά, η χρονική στιγμή που συνέβησαν).

Ερώτημα 1 (40%). Σχεσιακή Βάση Δεδομένων

a. Με βάση τα παραπάνω στοιχεία, σχεδιάστε το σχεσιακό σχήμα της ΒΔ, υλοποιήστε το (εντολές CREATE TABLE) στο ΣΔΒΔ PostgreSQL και φορτώστε με δεδομένα τους πίνακες. Ενδεχομένως να χρειαστεί να υλοποιήσετε επιπλέον βοηθητικούς πίνακες, σε σχέση με αυτούς οι οποίοι περιγράφονται στην εισαγωγή. Επιπλέον, καλείστε να τεκμηριώσετε τους περιορισμούς ακεραιότητας των πινάκων (και να δηλώσετε τυχόν περιορισμούς που προκύπτουν από την εκφώνηση αλλά δεν μπορέσατε να υποστηρίξετε μέσα από τους περιορισμούς ακεραιότητας των πινάκων). Το παραδοτέο του υποερωτήματος είναι το σχεσιακό σχήμα της ΒΔ, οι εντολές CREATE TABLE και τα αρχεία τα οποία θα εισάγετε στους πίνακες. Οδηγία: για την ευκολότερη παραγωγή αληθοφανών δεδομένων προτείνεται να χρησιμοποιήσετε κάποιο εργαλείο παραγωγής δεδομένων (data generator) (π.χ. www.mockaroo.com, https://faker.readthedocs.io/en/master/, https://devskiller.github.io/jfairy/).

b. Πάνω στο τελικό σχήμα της ΒΔ υλοποιήστε 2 προβολές/όψεις (views):

• Πρόγραμμα-αγώνων. Μια προβολή που θα αφορά μια συγκεκριμένη ημερομηνία (π.χ. 30/5/2023) και θα περιλαμβάνει τις «δυναμικές» πληροφορίες των αγώνων εκείνης της ημέρας: τόπος διεξαγωγής αγώνα, χρόνος, ποιες ομάδες συμμετέχουν, ποιο το σκορ, ποιοι παίκτες από κάθε ομάδα (όνομα θέση, στο παιχνίδι, χρόνος συμμετοχής στο παιχνίδι, τις κάρτες που τυχόν χρεώθηκε, τα γκολ που έβαλε και πότε τα έβαλε).

• Ετήσιο πρωτάθλημα αγώνων. Μια προβολή που θα αφορά μια συγκεκριμένη αγωνιστική σεζόν (π.χ. 1/9/2022 – 30/6/2023) και θα περιλαμβάνει τις «στατικές» πληροφορίες των αγώνων εκείνου του διαστήματος: τόπος διεξαγωγής αγώνα, χρόνος, ποιες ομάδες συμμετέχουν, ποιο το σκορ μεταξύ τους, ποια ομάδα είναι εντός/εκτός έδρας.

Ερώτημα 2 (20%). Εκτελέστε τις παρακάτω ερωτήσεις (queries) στη ΒΔ (εντολές SELECT).

a) Ποιος είναι προπονητής μιας συγκεκριμένης ομάδας σε συγκεκριμένο αγώνα;

b) Τα γκολ, πέναλτι που έγιναν σε συγκεκριμένο αγώνα, ποια χρονική στιγμή και από ποιόν παίκτη.

c) Την αγωνιστική εικόνα ενός συγκεκριμένου παίκτη για μια αγωνιστική σεζόν: γκολ, πέναλτι, κάρτες, λεπτά αγώνα, θέση που έπαιξε. d) Την αγωνιστική εικόνα μιας συγκεκριμένης ομάδας για μια αγωνιστική σεζόν: σε πόσους αγώνες συμμετείχε, σε πόσους ήταν γηπεδούχος και σε πόσους φιλοξενούμενη, πόσες ήττες /νίκες/ ισοπαλίες, πόσες φορές νίκησε/ έχασε/ έφερε ισοπαλία εντός/ εκτός έδρας.

Ερώτημα 3 (20%). Υλοποίηση triggers, cursors

a. Φτιάξτε έναν trigger ο οποίος κρατά/γεμίζει ένα πίνακα-ιστορικό. Όταν διαγράφονται με επιτυχία γραμμές από τον πίνακα ομάδες (π.χ. διαγράφονται όλες οι ομάδες οι οποίες δεν πέτυχαν καμία νίκη μέσα στο έτος) τότε οι διαγραμμένες γραμμές εισάγονται αυτόματα στον πίνακα ομάδες-υποβιβασμός-κατηγορίας.

b. Βρείτε για κάθε παίκτη ομαδοποιημένα ανά χρονικά διαστήματα και ανά ομάδα και ανά αγώνα τα: γκολ, πέναλτι, κάρτες, λεπτά αγώνα, θέση που έπαιξε. Χρησιμοποιείστε cursors ώστε να εμφανίσετε τις γραμμές σε ομάδες των 10.

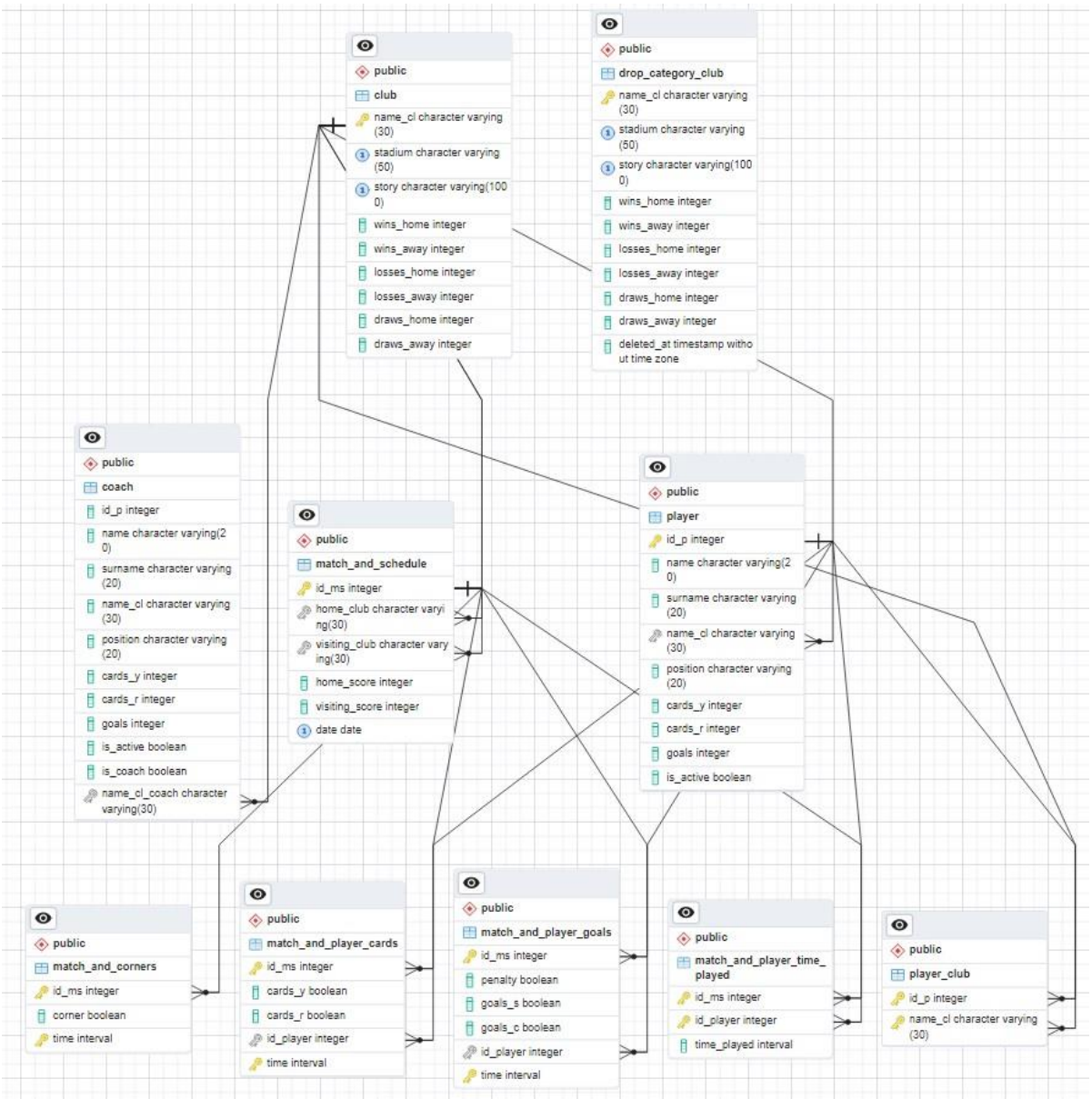
Ερώτημα 4 (20%). Σύνδεση ΒΔ με Application Programming Interface (API)

Υλοποιήστε προγραμματιστικά έναν client σε οποιαδήποτε γλώσσα προγραμματισμού γνωρίζετε (π.χ. Python, Java, C) χρησιμοποιώντας την κατάλληλη βιβλιοθήκη σύνδεσης με την PostgreSQL (π.χ. psycopg2, JDBC, ODBC). Ο client θα συνδέεται στο ΣΔΒΔ της PostgreSQL, θα εκτελεί τα queries του Ερωτήματος 2, και θα εμφανίζει τα αποτελέσματα στον χρήστη (είτε σε terminal είτε γραφικά).

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1.a. Εισαγωγή

1) Αφού πρώτα σκεφτήκαμε και σχεδιάσαμε στο χαρτί τι επιπλέον πίνακες θα χρειαστούμε από τους 4 που δίνονται στην εκφώνηση (Παίκτες, Προπονητές, Ομάδες, Αγώνες/πρόγραμμα αγώνων) καθώς και τα πεδία που θα έχουν , τα primary και foreign key, ξεκινήσαμε να γράφουμε τον SQL κώδικα για τα CREATE. Παρακάτω παραθέτουμε το ERD της βάσης μας ώστε να επεξηγήσουμε σύντομα το κάθε πεδίο που αποσκοπεί.



Πίνακας: club

Στήλες:

name_cl: VARCHAR(30) - Στήλη primary key που αντιπροσωπεύει το (μοναδικό) όνομα του συλλόγου.

stadium: VARCHAR(50) - Στήλη που αντιπροσωπεύει το όνομα του γηπέδου.

story: VARCHAR(1000) - Στήλη που αντιπροσωπεύει την σύντομη ιστορία του συλλόγου.

wins_home: INTEGER - Στήλη που αντιπροσωπεύει τον αριθμό των εντός έδρας νικών.

wins_away: INTEGER - Στήλη που αντιπροσωπεύει τον αριθμό των εκτός έδρας νικών.

losses_home: INTEGER - Στήλη που αντιπροσωπεύει τον αριθμό των εντός έδρας ηττών.

losses_away: INTEGER - Στήλη που αντιπροσωπεύει τον αριθμό των εκτός έδρας ηττών.

draws_home: INTEGER - Στήλη που αντιπροσωπεύει τον αριθμό των εντός έδρας ισοπαλιών.

draws_away: INTEGER - Στήλη που αντιπροσωπεύει τον αριθμό των εκτός έδρας ισοπαλιών.

Πίνακας: player

Στήλες:

id_p: SERIAL - Στήλη primary key που αντιπροσωπεύει το μοναδικό αναγνωριστικό του παίκτη. **(SERIAL σημαίνει ότι όταν κάνουμε insert έναν παίχτη δεν θα χρειάζεται να δίνουμε και id_p, αφού αυτό θα μπαίνει μόνο του ξεκινώντας από το 1 και προσθέτοντας +1 σε κάθε insert.**

name: VARCHAR(20) - Στήλη που αντιπροσωπεύει το πρώτο όνομα του παίκτη.

surname: VARCHAR(20) - Στήλη που αντιπροσωπεύει το επώνυμο του παίκτη.

name_cl: VARCHAR(30) - Στήλη foreign key στο name_cl του πίνακα club που αναφέρεται στο σωματείο στο οποίο ανήκει ο παίκτης. **(foreign key επειδή το name_cl του player «παίρνει τιμή» από το name_cl του club)**

position: VARCHAR(20) - Στήλη που αντιπροσωπεύει τη θέση του παίκτη σε όλη του την καριέρα. (Θεωρούμε ότι είναι μόνο μία και δεν αλλάζει)

cards_y: INTEGER - Στήλη που αντιπροσωπεύει τον αριθμό των κίτρινων καρτών που δέχθηκε ο παίκτης σε όλη του την καριέρα.

cards_r: INTEGER - Στήλη που αντιπροσωπεύει τον αριθμό των κόκκινων καρτών που έλαβε ο παίκτης σε όλη του την καριέρα.

goals: INTEGER - Στήλη που αντιπροσωπεύει τον αριθμό των τερμάτων που σημείωσε ο παίκτης σε όλη του την καριέρα.

is_active: BOOLEAN - Στήλη που αντιπροσωπεύει αν ο παίκτης είναι ενεργός ακόμα και σήμερα.

Πίνακας: player_club (πίνακας ιστορικών που δείχνει που έχει παίξει ένας παίχτης δηλαδή μεταγραφές)

Στήλες:

id_p: INTEGER - Στήλη foreign key στο id_p του πίνακα player που αναφέρεται στο αναγνωριστικό του παίκτη.

name_cl: VARCHAR(30) - Στήλη foreign key στο name_cl του πίνακα club που αναφέρεται στο σωματείο στο οποίο έχει υπάρξει ένας παίκτης.

Πρωτεύον κλειδί: (id_p, name_cl) – Ο συνδυασμός τους.

Πίνακας: προπονητής (κληρονομεί από τον παίκτη αφού έχουν ίδια πεδία + μερικά ακόμα ο coach)

Στήλες:

is_coach: BOOLEAN - Στήλη που αντιπροσωπεύει αν ο παίκτης είναι προπονητής.

name_cl_coach: VARCHAR(30) - Στήλη foreign key που αναφέρεται στο σωματείο στο οποίο ανήκει ο προπονητής τώρα. **(Υποθέτουμε πως ένας προπονητής δεν μπορεί να κάνει μεταγραφές όπως ο player, ενναλακτικά θα είχαμε φτιάξει έναν πίνακα ιστορικού όπως στον player)**

Πίνακας: match_and_schedule

Στήλες:

id_ms: SERIAL - Στήλη primary key που αντιπροσωπεύει το μοναδικό αναγνωριστικό του αγώνα. (Αναλύσαμε πριν τι κάνει το SERIAL)

home_club: VARCHAR(30) - Στήλη foreign key στο name_cl του πίνακα club που αναφέρεται στον σύλλογο έδρας.

visiting_club: VARCHAR(30) - Στήλη foreign key στο name_cl του πίνακα club που αναφέρεται στον φιλοξενούμενο σύλλογο.

home_score: INTEGER - Στήλη που αντιπροσωπεύει το σκορ της γηπεδούχου ομάδας.

visiting_score: INTEGER - Στήλη που αντιπροσωπεύει το σκορ της φιλοξενούμενης ομάδας.

date: DATE - Στήλη που αντιπροσωπεύει την ημερομηνία του αγώνα π.χ. 23/05/2023.

Πίνακας: match_and_player_goals

Στήλες:

id_ms: INTEGER - Στήλη foreign key στο id_ms του πίνακα match_and_schedule που αναφέρεται στο αναγνωριστικό του αγώνα.

penalty: BOOLEAN - Στήλη που αντιπροσωπεύει αν εκτελέστηκε penalty με true και το χρονικό λεπτό που έγινε.

goals_s: BOOLEAN - Στήλη που αντιπροσωπεύει ένα γκολ που μπήκε από κάποιον παίχτη και ποιο λεπτό.

goals_c: BOOLEAN - Στήλη που αντιπροσωπεύει αν μπήκε κάποιο γκολ και ακυρώθηκε από κάποιον παίχτη και ποιο λεπτό .

id_player: INTEGER - Στήλη foreign key στο id_p του πίνακα player που αναφέρεται στον παίκτη που εκτέλεσε penalty ή έβαλε γκολ, ή του ακυρώθηκε το γκολ. **(Επειδή μετράμε δευτερόλεπτα σε αγώνα ποδοσφαίρου, δεν μπορεί να γίνουν ταυτόχρονα κάποια από αυτά)**

time: INTERVAL – Κρατάμε τον χρόνο π.χ. στο 68:23 που έγινε κάτι από τα παραπάνω.

Πρωτεύον κλειδί: (id_ms, time) – Ο συνδυασμός τους .

Πίνακας: match_and_player_corners

Στήλες:

id_ms: INTEGER - Στήλη foreign key στο id_ms του πίνακα match_and_schedule που αναφέρεται στο αναγνωριστικό του αγώνα.

corner: BOOLEAN - Στήλη που αντιπροσωπεύει αν εκτελέστηκε corner με true και το χρονικό λεπτό που έγινε.

time: INTERVAL – Κρατάμε τον χρόνο π.χ. στο 68:23 που έγινε corner

Πρωτεύον κλειδί: (id_ms, time) – Ο συνδυασμός τους .

Πίνακας: match_and_player_cards

Στήλες:

id_ms: INTEGER - Στήλη foreign key στο id_ms του πίνακα match_and_schedule που αναφέρεται στο αναγνωριστικό του αγώνα.

cards_y: BOOLEAN - Στήλη που αντιπροσωπεύει αν πήρε κίτρινη κάρτα με true και το χρονικό λεπτό που έγινε και ποιος την πήρε.

cards_r: BOOLEAN - Στήλη που αντιπροσωπεύει αν πήρε κόκκινη κάρτα με true και το χρονικό λεπτό που έγινε και ποιος την πήρε.

id_player: INTEGER - Στήλη foreign key στο id_p του πίνακα player που αναφέρεται στον παίκτη που πήρε κίτρινη ή κόκκινη κάρτα. **(Επειδή μετράμε δευτερόλεπτα σε αγώνα ποδοσφαίρου, δεν μπορεί να γίνουν ταυτόχρονα κάποια από αυτά)**

time: INTERVAL – Κρατάμε τον χρόνο π.χ. στο 68:23 που έγινε κάτι από τα παραπάνω.

Πρωτεύον κλειδί: (id_ms, time) – Ο συνδυασμός τους .

Πίνακας: match_and_player_time_played

Στήλες:

id_ms: INTEGER - Στήλη foreign key στο id_ms του πίνακα match_and_schedule που αναφέρεται στο αναγνωριστικό του αγώνα.

id_player: INTEGER - Στήλη foreign key στο id_p του πίνακα player που αναφέρεται πόση ώρα έπαιξε σε ένα ματς ένας παίχτης. **(Υποθέτουμε πως δεν βγαίνει και ξαναμπαίνει κάποιος μιας και σύμφωνα με την εκφώνηση έχουμε 11 παίχτες για κάθε ομάδα άρα δεν έχουμε αλλαγές)**

time_played: INTEGER – Κρατάμε τον χρόνο π.χ. 68:23 που έπαιξε ένας παίχτης σε έναν αγώνα.

Πρωτεύον κλειδί: (id_ms, id_player) – Ο συνδυασμός τους .

2) Επόμενο, βήμα ήταν η εισαγωγή δεδομένων καθώς και η δημιουργία περιορισμών που προκύπτουν από την εκφώνηση.

Για να μην επαναλαμβανόμαστε συνέχεια, δεν θα παραθέσουμε όλα τα insert, μόνα όσα έχουμε να σημειώσουμε κάτι .

```
INSERT INTO player (id_p, name, surname, name_cl, position, cards_y, cards_r, goals, is_active)
VALUES
        (DEFAULT, 'Παναγιώτης', 'Ρέτσος', 'Olympiacos', 'Defender', 2, 0, 0, false),
        (DEFAULT, 'Κώστας', 'Φορτούνης', 'Olympiacos', 'Midfielder', 0, 0, 2, true),
        (DEFAULT, 'Υούσεφ', 'Ελ-Αραμπί', 'Olympiacos', 'Forward', 0, 0, 1, true),
        (DEFAULT, 'Τζέιμς', 'Ροντρίγκες', 'Olympiacos', 'Midfielder', 0, 0, 0, true),
        (DEFAULT, 'Μαρσέλο', 'Βιέιρα', 'Olympiacos', 'Defender', 0, 0, 2, true),
        (DEFAULT, 'Γιώργος', 'Τζαβέλας', 'Aek', 'Defender', 0, 0, 1, false),
        (DEFAULT, 'Πέτρος', 'Μάνταλος', 'Aek', 'Forward', 0, 0, 8, true),
        (DEFAULT, 'Γιώργος', 'Αθανασιάδης', 'Aek', 'GoalKeeper', 1, 0, 2, true),
        (DEFAULT, 'Κωνσταντίνος', 'Γαλανόπουλος', 'Aek', 'Midfielder', 0, 0, 7, true),
        (DEFAULT, 'Λάζαρος', 'Ρότας', 'Aek', 'Defender', 0, 0, 2, true),
        (DEFAULT, 'Φώτης', 'Ιωαννίδης', 'Panathinaikos', 'Forward', 0, 0, 12, true),
        (DEFAULT, 'Λεονάρντο', 'Φρόκκου', 'Panathinaikos', 'Midfielder', 0, 1, 0, true),
        (DEFAULT, 'Αλμπέρτο', 'Μπρινιόλι', 'Panathinaikos', 'Goalkeeper', 0, 0, 12, true),
        (DEFAULT, 'Σεμπαστιάν', 'Παλάσιος', 'Panathinaikos', 'Midfielder', 0, 0, 12, true),
        (DEFAULT, 'Ρούμπεν', 'Πέρεθ', 'Panathinaikos', 'Midfielder', 0, 0, 12, false);
```

Για λόγους ευκολίας στην εισαγωγή δεδομένων βάλαμε σε κάθε ομάδα 5 παίχτες αντί για 11.

Με πέρασμα της τιμής DEFAULT στο SERIAL πεδίο id_p, αφήνουμε την POSTGRESQL μόνη της να βάζει σειριακά τα id_p π.χ. Ο Παναγιώτης Ρέτσος έχει id_p = 1, ο Κώστας Φουρτούνης id_p = 2 και τα λοιπά.

INSERT INTO coach (id_p, name, surname, name_cl, position, cards_y, cards_r, goals, is_active, is_coach, name_cl_coach)

SELECT id_p, name, surname, name_cl, position, cards_y, cards_r, goals, is_active, true, name_cl

FROM player

WHERE id_p = (SELECT id_p FROM player WHERE name = 'Παναγιώτης' AND surname = 'Ρέτσος' AND player.is_active = false);

INSERT INTO coach (id_p, name, surname, name_cl, position, cards_y, cards_r, goals, is_active, is_coach, name_cl_coach)

SELECT id_p, name, surname, name_cl, position, cards_y, cards_r, goals, is_active, true, name_cl

FROM player

WHERE id_p = (SELECT id_p FROM player WHERE name = 'Ρούμπεν' AND surname = 'Πέρεθ' AND player.is_active = false);

Η εισαγωγή στον coach γίνεται με αυτόν τον τρόπο αφού ένας coach δεν είναι κάτι εντελώς καινούργιο, αλλά ένας player που πλέον δεν είναι active και έχει 2 καινούργια πεδία τα is_coach, name_cl_coach. Για αυτό και δεν περνάμε καινούργιες τιμές για τα υπόλοιπα πεδία, αλλά τα παίρνουμε έτοιμα από κάποιον player που έχουμε πει εμείς με την συνθήκη WHERE.

Όσο αφορά τους περιορισμούς,

υλοποιήσαμε το να είναι ένα club name, ένα στάδιο και μια ιστορία μοναδικά (για να ταυτίζονται με την πραγματικότητα) με το keyword: UNIQUE στα αντίστοιχα πεδία όταν δημιουργήσαμε τον πίνακα club.

Βάλαμε constraints:

i) Στον πίνακα match_and_schedule, { (CONSTRAINT unique_match_teams_date UNIQUE (home_club, visiting_club, date) } ώστε να εμποδίσουμε ένα 2 ομάδες να παίξουν ξανά την ίδια μέρα.

ii) Στον πίνακα player, { CONSTRAINT name_charset CHECK (name ~ '^[A-Ωα-ωάέήίϊΐόύϋΰώ\-]+$' AND surname ~ '^[A-Ωα-ωάέήίϊΐόύϋΰώ\-]+$') } ώστε να εμποδίσουμε την εισαγωγή ονόματος που δεν είναιγραμμένο στα ελληνικά, όπως ήθελε η εκφώνηση. Ειδικότερα για να το πετύχουμε αυτό χρησιμοποιήσαμε Regular

Expressions, όπου '^' δηλώνει την αρχή του string , και '$' το τέλος του, το οποίο string δέχεται ελληνικούς χαρακτήρες από το Α έως το Ω, φωνήεντα με τόνους ή διαλυτικά και παύλα (-).

iii) Καθώς και functions που καλούνται με triggers σε κάποιο insert, με αποτέλεσμα να προλαβαίνουν να μην εισαχθούν είτε πάνω από 11 παίχτες σε μια ομάδα είτε κάποια ομάδα να παίζει σε διάστημα μικρότερο από 10 μέρες. Αντίστοιχα τα functions, triggers:

```
CREATE OR REPLACE FUNCTION check_player_count()
 RETURNS TRIGGER AS
$$
DECLARE
 player_count INTEGER;
BEGIN
 -- Get the count of rows for the current club name
 SELECT COUNT(*) INTO player_count
 FROM player
 WHERE name_cl = NEW.name_cl;

 -- Raise an exception if the row count exceeds the limit
 IF player_count >= 11 THEN
  RAISE EXCEPTION 'Maximum row limit reached for club: %', NEW.name_cl;
 END IF;
 RETURN NEW;
END;
$$
LANGUAGE plpgsql;



-- Create a trigger that calls the check_player_count function before insert
CREATE TRIGGER limit_player_count
BEFORE INSERT ON player
FOR EACH ROW
EXECUTE FUNCTION check_player_count();
```

```
CREATE OR REPLACE FUNCTION check_match_schedule() RETURNS TRIGGER AS $$
DECLARE
   previous_match_date DATE;
BEGIN
   -- Check if the home team has a match within 10 days before or after the current match
   SELECT date INTO previous_match_date
   FROM match_and_schedule
   WHERE (home_club = NEW.home_club OR visiting_club = NEW.home_club)
   AND date >= NEW.date - INTERVAL '10 days'
   AND date <= NEW.date + INTERVAL '10 days'
   AND id_ms != NEW.id_ms;

   IF previous_match_date IS NOT NULL THEN
      RAISE EXCEPTION 'A team has a match within 10 days before or after this match.';
   END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;


-- Create a trigger to call the function before inserting into the match_and_schedule table
CREATE TRIGGER check_match_schedule_trigger
BEFORE INSERT ON match_and_schedule
FOR EACH ROW
EXECUTE FUNCTION check_match_schedule();
```

1.c. Views

Η άσκηση ζητά 2 απλά views (δηλαδή 2 Selects που αποθηκεύονται ώστε να μπορούμε να τα καλούμε όποτε θέλουμε)

a) CREATE VIEW schedule_match AS

SELECT match_and_schedule.date,match_and_schedule.home_club,
match_and_schedule.visiting_club,match_and_schedule.home_score,
match_and_schedule.visiting_score,club.stadium,player.name,player.surname, player.position,
player.name_cl,match_and_player_cards.cards_y,match_and_player_cards.cards_r,
match_and_player_time_played.time_played,match_and_player_goals.goals_s,
match_and_player_goals.time

FROM match_and_schedule

FULL JOIN club ON match_and_schedule.home_club = club.name_cl

FULL JOIN player ON match_and_schedule.home_club = player.name_cl OR
match_and_schedule.visiting_club = player.name_cl

LEFT JOIN match_and_player_cards ON player.id_p = match_and_player_cards.id_player AND
match_and_schedule.id_ms = match_and_player_cards.id_ms

FULL JOIN match_and_player_goals ON match_and_schedule.id_ms =
match_and_player_goals.id_ms AND match_and_player_goals.id_player = player.id_p

FULL JOIN match_and_player_time_played ON match_and_player_time_played.id_ms =
match_and_schedule.id_ms AND match_and_player_time_played.id_player = player.id_p

WHERE match_and_schedule.date = '2023-06-29' AND player.is_active = true;

Στο SELECT βάζουμε τα πεδία που μας ζητούνται να φαίνονται. Στο From τον βασικό πίνακα από όπου παίρνουμε κάποια από τα δεδομένα του SELECT τον οποίο με FULL JOIN, LEFT JOIN ενώνουμε με τους άλλους πίνακες που χρειαζόμαστε για τα πεδία του SELECT, ενώνοντάς τους ON ένα κοινό πεδίο. Στο WHERE μπαίνει η συνθήκη μας όπου εδώ θέλουμε μια συγκεκριμένη ημερομηνία.

b) CREATE VIEW season_matches AS SELECT match_and_schedule.date,
match_and_schedule.home_club, match_and_schedule.visiting_club,
match_and_schedule.home_score, match_and_schedule.visiting_score, club.stadium

FROM match_and_schedule JOIN club ON match_and_schedule.home_club = club.name_cl
WHERE match_and_schedule.date BETWEEN '2023-04-01' AND '2023-06-30' ORDER BY date;

SELECT * FROM season_matches; ΠΑΡΟΜΟΙΑ ΚΑΙ ΕΔΩΠεριγραφή του προγράμματος

2 Εκτελέστε τις παρακάτω ερωτήσεις (queries) στη ΒΔ (εντολές SELECT).

2.a) SELECT club.name_cl, coach.name, coach.surname FROM match_and_scheduleJOIN club ON (match_and_schedule.home_club = club.name_cl OR match_and_schedule.visiting_club = club.name_cl) JOIN coach ON (club.name_cl = coach.name_cl AND coach.is_coach = true) WHERE match_and_schedule.id_ms = 10 AND club.name_cl = 'Olympiacos' AND coach.is_coach = true;

Για να δείξουμε ποιος είναι προπονητής μιας συγκεκριμένης ομάδας σε συγκεκριμένο αγώνα, κάνουμε select τα club.name_cl, coach.name, coach.surname από τον match_and_schedule επειδή θέλουμε για έναν συγκεκριμένο αγώνα, ο οποίος ενώνεται με τον club με το κοινό τους πεδίο match_and_schedule.home_club = club.name_cl ή match_and_schedule.visiting_club = club.name_cl. Ο οποίος με την σειρά ενώνεται με τον coach στο club.name_cl = coach.name_cl για να πάρουμε τα coach.name, coach.surname. Στην συνθήκη WHERE μπαίνει το όνομα ενός συγκεκριμένου κλαμπ, το id ενός συγκεκριμένου αγώνα καθώς και coach.is_coach = true για να πάρουμε τον προπονητή μόνο.

2.b) SELECT match_and_player_goals.penalty, match_and_player_goals.goals_s, match_and_player_goals.time, player.name, player.surname
FROM match_and_schedule JOIN match_and_player_goals ON match_and_player_goals.id_ms = match_and_schedule.id_ms JOIN player ON player.id_p = match_and_player_goals.id_player
WHERE match_and_schedule.id_ms = 10

Ακριβώς ίδια λογική με το προηγούμενο, δεν έχουμε να σημειώσουμε κάτι

2.c) SELECT player.id_p, player.name, player.surname, player.position,
SUM(match_and_player_time_played.time_played) AS total_time_played,
COALESCE(SUM(CASE WHEN match_and_player_goals.goals_s = true THEN 1 ELSE 0 END),0) AS player_goals,
COALESCE(SUM(CASE WHEN match_and_player_goals.penalty = true THEN 1 ELSE 0 END),0) AS player_penalties,
COALESCE(SUM(CASE WHEN match_and_player_cards.cards_y = true THEN 1 ELSE 0 END),0) AS player_yellow_cards,
COALESCE(SUM(CASE WHEN match_and_player_cards.cards_r = true THEN 1 ELSE 0 END),0) AS player_red_cards

FROM match_and_schedule
NATURAL JOIN match_and_player_time_played
JOIN player ON player.id_p = match_and_player_time_played.id_player
NATURAL JOIN match_and_player_goals

```
LEFT JOIN match_and_player_cards ON match_and_player_cards.id_player =
match_and_player_goals.id_player
WHERE (match_and_schedule.date BETWEEN '2023-04-01' AND '2023-06-30') AND
match_and_player_time_played.id_player = 14
GROUP BY player.id_p
```

Με το SUM(match_and_player_time_played.time_played)AS total_time_played, χρησιμοποιούμε την συνάρτηση SUM για να αθροίσουμε τον χρόνο που έχει παίξει ο παίχτης στους αγώνες μεταξύ 2023-04-01 και 2023-06-30, και αποθηκεύουμε τη τιμή σε ένα καινούργιο πεδίο:total_time_played

Με το COALESCE(SUM(CASE WHEN match_and_player_goals.goals_s = true THEN 1 ELSE 0 END),0) AS player_goals, υπολογίζουμε το άθροισμα των γκολ που σημείωσε ο παίκτης ελέγχοντας τη στήλη goals_s στον πίνακα match_and_player_goals. Επειδή το πεδίο είναι Boolean, εάν η συνθήκη είναι true, προσθέτει 1, διαφορετικά προσθέτει 0. Η συνάρτηση COALESCE χρησιμοποιείται για να αντικαταστήσει τα αποτελέσματα NULL-false με 0 και το αποτέλεσμα έχει ψευδώνυμο player_goals.

Το NATURAL JOIN δεν χρειάζεται να ορίσεις τα πεδία που θα συνδεθούν αυτόματα συνδέει αυτά που έχουν ίδιο όνομα.

2.d)

```
SELECT home_matches, away_matches,  total_home_score,  total_away_score,

 total_home_score + total_away_score AS total_score,  home_matches + away_matches AS
total_matches, subquery.total_wins_home,subquery.total_wins_away, subquery.total_wins_home
+ subquery.total_wins_away AS total_wins,
subquery.total_draws_home,subquery.total_draws_away   subquery.total_draws_home +
subquery.total_draws_away AS total_draws, (home_matches + away_matches) -
(subquery.total_wins_home + subquery.total_wins_away + subquery.total_draws_home +
subquery.total_draws_away) AS total_losses, home_matches - (subquery.total_wins_home +
subquery.total_draws_home) AS total_losses_home, away_matches - (subquery.total_wins_away +
subquery.total_draws_away) AS total_losses_away
FROM (SELECT
SUM(CASE WHEN match_and_schedule.home_club='Aek' THEN 1 ELSE 0 END) AS home_matches,
SUM(CASE WHEN match_and_schedule.visiting_club='Aek' THEN 1 ELSE 0 END) AS away_matches,
SUM(CASE WHEN match_and_schedule.home_club='Aek' THEN match_and_schedule.home_score
ELSE 0 END) AS total_home_score,
SUM(CASE WHEN match_and_schedule.visiting_club='Aek' THEN
match_and_schedule.visiting_score ELSE 0 END) AS total_away_score,
SUM(CASE WHEN (match_and_schedule.home_club='Aek' AND (match_and_schedule.home_score
> match_and_schedule.visiting_score)) THEN 1 ELSE 0 END) AS total_wins_home,
SUM(CASE WHEN (match_and_schedule.visiting_club='Aek' AND
(match_and_schedule.visiting_score > match_and_schedule.home_score)) THEN 1 ELSE 0 END) AS
total_wins_away,

 SUM(CASE WHEN (match_and_schedule.home_club='Aek' AND (match_and_schedule.home_score
= match_and_schedule.visiting_score)) THEN 1 ELSE 0 END) AS total_draws_home,
SUM(CASE WHEN (match_and_schedule.visiting_club='Aek' AND
(match_and_schedule.visiting_score = match_and_schedule.home_score)) THEN 1 ELSE 0 END) AS
```

```
total_draws_away
FROM match_and_schedule
WHERE match_and_schedule.date BETWEEN '2023-04-01' AND '2023-06-30' AS subquery;
```

Για την υλοποίηση χρησιμοποιήθηκαν 2 SELECT statements. Το ένα βρίσκεται μέσα στο FROM του 1ου. Αυτό ονομάζεται subquery. Το υποερώτημα χρησιμοποιείται σε αυτό το ερώτημα SQL για τον υπολογισμό των συγκεκριμένων στατιστικών που αφορούν την ομάδα "Aek" για το συγκεκριμένο εύρος ημερομηνιών. Το υποερώτημα είναι ουσιαστικά ένα εμφωλευμένο ερώτημα που εκτελείται πρώτα και δημιουργεί ένα προσωρινό σύνολο αποτελεσμάτων. Χρησιμοποιώντας ένα υποερώτημα, μπορείτε να εκτελέσετε αθροίσεις και υπολογισμούς στον πίνακα match_and_schedule ειδικά για την ομάδα "Aek" και το καθορισμένο εύρος ημερομηνιών. Αυτό σας επιτρέπει να λάβετε τις απαραίτητες πληροφορίες για το τελικό σύνολο αποτελεσμάτων του εξωτερικού ερωτήματος. Το υποερώτημα υπολογίζει στατιστικά στοιχεία όπως ο αριθμός των εντός έδρας αγώνων, των εκτός έδρας αγώνων, το συνολικό σκορ εντός έδρας, το συνολικό σκορ εκτός έδρας, τις νίκες και τις ισοπαλίες για την ομάδα. Αυτές οι υπολογισμένες τιμές χρησιμοποιούνται στη συνέχεια στο εξωτερικό ερώτημα για την εξαγωγή πρόσθετων στατιστικών στοιχείων και την παρουσίαση μιας ολοκληρωμένης επισκόπησης της απόδοσης της ομάδας. Συνοπτικά, το υποερώτημα βοηθά στην απομόνωση και τον υπολογισμό των στατιστικών στοιχείων που αφορούν την ομάδα εντός ενός συγκεκριμένου εύρους ημερομηνιών, επιτρέποντας στο εξωτερικό ερώτημα να παράγει το επιθυμητό σύνολο αποτελεσμάτων με συγκεντρωτικές πληροφορίες.

3 Υλοποίηση triggers, cursors

3.a) Αρχικά, όπως αναφέρεται και στην εκφώνηση, το αποτέλεσμα του trigger που ζητείται να υλοποιηθεί, πρέπει να εισάγονται σε έναν πίνακα 'drop_category_club', ο οποίος περιέχει ακριβώς τα ίδια πεδία με τον πίνακα clubs και ένα επιπλέον πεδίο το deleted_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP. Αυτό το πεδίο εισάχθηκε, καθώς θέλουμε να κρατάμε πότε έγινε delete ένα πεδίο (κρατάει την ημερομηνία), ενώ είναι null εάν η γραμμή αυτή δεν προέκυψε από διαγραφή.

Στη συνέχεια, κατασκευάσαμε την συνάρτηση 'clubs_log_function()', που εκτελείται κάθε φορά που πυροδοτείται ο trigger. Η συνάρτηση είναι σε PL/pgSQL και εισάγει στον πίνακα 'drop_category_club' της μόλις διαγραμμένες τιμές της αντίστοιχης στήλης μέσω της χρήσης του OLD keyword.

Ύστερα, κατασκευάζουμε και τον trigger, ο οποίος ενεργοποιείται κάθε φορά μετά (AFTER) από μία διαγραφή του πίνακα club και εκτελεί την παραπάνω συνάρτηση.

```
create table drop_category_club
( name_cl VARCHAR(30)  PRIMARY KEY UNIQUE,
  stadium VARCHAR(50) NOT NULL UNIQUE,
 story VARCHAR(1000) NOT NULL UNIQUE,
 wins_home INTEGER NOT NULL,
 wins_away INTEGER NOT NULL,
 losses_home INTEGER NOT NULL,
 losses_away INTEGER NOT NULL,
 draws_home INTEGER NOT NULL,
 draws_away INTEGER NOT NULL,
 deleted_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP);
--valame to deleted_at giati ayto enhmerwnetai kathe fora pou ena pedio ginetai DELETE



CREATE FUNCTION clubs_log_function() RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO drop_category_club (name_cl, stadium, story, wins_home, wins_away, losses_home, losses_away, draws_home, draws_away)
    VALUES (OLD.name_cl, OLD.stadium, OLD.story, OLD.wins_home, OLD.wins_away, OLD.losses_home, OLD.losses_away, OLD.draws_home, OLD.draws_away);
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER clubs_log
AFTER DELETE ON club
FOR EACH ROW
EXECUTE FUNCTION clubs_log_function();

DELETE FROM club WHERE name_cl = 'PAOK'

INSERT INTO club (name_cl, stadium, story, wins_home, wins_away, losses_home, losses_away, draws_home, draws_away)
VALUES ('PAOK', 'TOUMPA ARENA', 'story.....', 2, 0, 1, 1, 1);

SELECT * FROM drop_category_club
```

3.b) Πρώτα, επικεντρωθήκαμε στο τι πρέπει να επιστρέψουμε (ποια πεδία) όταν θα εκτελείται ο cursor. Επομένως αξιοποιήσαμε το ήδη υπάρχον view του ερωτήματος 1.c (αφού είχε τα πεδία που χρειαζόμασταν: γκολ, πέναλτι, κάρτες, λεπτά αγώνα, θέση που έπαιξε) και τον ονομάσαμε 'cursor_view' και η μοναδική αλλαγή που εφαρμόσαμε είναι στην ημερομηνία στην συνθήκη WHERE όπου βάλαμε μεταξύ της πρώτη και της τελευταίας από αυτές που περιείχαν τα δεδομένα στους πίνακες, αλλά και στο SELECT προσθέσαμε το πεδίο 'match_and_schedule.id_ms' (id του αγώνα).

Όσον αφορά τον cursor αυτόν καθαυτών υλοποιήθηκε σε PL/pgSQL. Πρώτα, δημιουργούμε τον cursor group_cursor, ο οποίος κάνει SELECT δεδομένα από το παραπάνω

view, και χρησιμοποιώντας την COUNT, υπολογίζει τα γκολ, πέναλτι, κίτρινες κάρτες, κόκκινες κάρτες αλλά επιλέγει και τα πεδία λεπτά αγώνα, θέση που έπαιξε και την ημερομηνία και τα κάνουμε GROUP BY την ημερομηνία, την ομάδα αλλά και τον αγώνα (τα υπόλοιπα GROUP BY τα προσθέσαμε καθώς η PostgreSQL απαιτεί όλα τα πεδία τα οποία βρίσκονται εκτός aggregate function να εμφανίζονται στο GROUP BY).

Συνεχίζοντας, ορίζουμε την μεταβλητή ‘current_row’ τύπου RECORD, η οποία μπορεί να αποθηκεύσει μια σειρά δεδομένων με μεταβαλλόμενη δομή, γραμμή η οποία ανακτήθηκε από τον cursor και στη συνέχεια θα έχουμε πρόσβαση στο κάθε πεδίο ξεχωριστά σε κάθε επανάληψη του βρόχου. Μετά, ορίζουμε την ακέραια μεταβλητή ‘counter’ η οποία καταμετράει τις διατρέξεις.

Αφότου, σηματοδοτούμε την έναρξη ενός μπλοκ κώδικα με το BEGIN και κάνουμε OPEN τον cursor για ανάκτηση γραμμών. Αρχίζουμε μετά έναν βρόχο που κάνει ανάκτηση (fetch) γραμμών μέχρι να τελειώσουν οι εγγραφές του πίνακα (NOT FOUND). Σε κάθε επανάληψη, κάνουμε RAISE NOTICE (τα δείχνουμε στην κονσόλα) τα ζητούμενα (προαναφερθείσα) πεδία, με χρήση της μεταβλητής current_row και του αντίστοιχου πεδίου (current_row.’column’). Επιπλέον, σε κάθε επανάληψη ο counter αυξάνεται κατά 1. Τέλος, επειδή ζητείται να εμφανίζεται το αποτέλεσμα σε ομάδες των 10 χρησιμοποιούμε έναν διαχωριστή (separator), που δηλώνεται με τη χρήση παυλών (----------).

```
DO $$
DECLARE
    group_cursor CURSOR FOR
        SELECT COUNT(subquery.goals_s), COUNT(subquery.penalty), COUNT(subquery.cards_r), COUNT(subquery.cards_y), subquery.time_played, subquery.position, subquery.date
        FROM (SELECT match_and_schedule.id_ms, match_and_schedule.date, match_and_schedule.home_club, match_and_schedule.visiting_club, match_and_schedule.home_score, match_and_schedule.visiting_score,
        club.stadium, player.name, player.surname, player.position, player.name_cl, match_and_player_cards.cards_y ,match_and_player_cards.cards_r,  match_and_player_time_played.time_played,
        match_and_player_goals.goals_s, match_and_player_goals.penalty, match_and_player_goals.time
        FROM
                        match_and_schedule
                 FULL JOIN
                        club ON match_and_schedule.home_club = club.name_cl
                FULL JOIN
                          player ON match_and_schedule.home_club = player.name_cl OR match_and_schedule.visiting_club = player.name_cl
                LEFT  JOIN
                        match_and_player_cards ON player.id_p = match_and_player_cards.id_player AND match_and_schedule.id_ms = match_and_player_cards.id_ms
                FULL JOIN
                        match_and_player_goals ON match_and_schedule.id_ms = match_and_player_goals.id_ms AND match_and_player_goals.id_player = player.id_p
                FULL JOIN
                          match_and_player_time_played ON match_and_player_time_played.id_ms = match_and_schedule.id_ms AND match_and_player_time_played.id_player = player.id_p
        WHERE
                        (match_and_schedule.date BETWEEN ('2023-3-10') AND  ('2023-06-15')) AND
                        player.is_active = true ) AS subquery
        GROUP BY subquery.date, subquery.home_club, subquery.id_ms, subquery.time_played,subquery.position, subquery.date  ;

    current_row RECORD;
    counter INTEGER := 0;
BEGIN
    OPEN group_cursor;

    LOOP
        FETCH group_cursor INTO current_row;
        EXIT WHEN NOT FOUND;

        RAISE NOTICE 'Goals: %, Penalties: %, Cards R: %, Cards Y: %, Time Played: %, Position: %, Date: %',
            current_row.count, current_row.count, current_row.count, current_row.count, current_row.time_played, current_row.position, current_row.date ;
        counter := counter + 1;

        IF counter = 10 THEN
            counter := 0;
            RAISE NOTICE '----------';
        END IF;
    END LOOP;

    CLOSE group_cursor;
END;
$$;
```

4. Σύνδεση ΒΔ με Application Programming Interface (API)

Σύνδεση ΒΔ με Application Programming Interface (API)

Ο κώδικας μπορεί να βρεθεί στο: \FootballCup\src\com\unipi\FootballCup\Main.java

Εκτελεί τα 4 ερωτήματα SQL του 2ου ερωτήματος της άσκησης για να λάβει τα διάφορα στατιστικά στοιχεία σχετικά με ποδοσφαιρικούς αγώνες, συλλόγους, παίκτες και προπονητές.

Πώς λειτουργεί:

1. Δημιουργεί μια σύνδεση με τη βάση δεδομένων FootballCup χρησιμοποιώντας τον JDBC jar connector που κατεβάσαμε και βάλαμε στον φάκελο lib. Επίσης δημιουργεί ένα αντικείμενο Statement για την εκτέλεση εντολών SQL.

```
try {
    Connection connection = DriverManager.getConnection( url: "jdbc:postgresql://localhost:5432/FootballCup",  user: "postgres",  password: "alexhs123");
    Statement statement = connection.createStatement();
```

2.SELECT 2.a:

- Εκτελεί το ερώτημα 2α για να ανακτήσει το όνομα του συλλόγου, το όνομα του προπονητή και το επώνυμο του προπονητή για έναν συγκεκριμένο αγώνα και σύλλογο.

- Εκτυπώνει τα δεδομένα που ανακτήθηκαν.

```
// Selection 2.a
String query2a = "SELECT club.name_cl, coach.name, coach.surname " +
        "FROM match_and_schedule " +
        "JOIN club ON (match_and_schedule.home_club = club.name_cl OR match_and_schedule.visiting_club = club.name_cl) " +
        "JOIN coach ON (club.name_cl = coach.name_cl AND coach.is_coach = true) " +
        "WHERE match_and_schedule.id_ms = 10 AND club.name_cl = 'Olympiacos' AND coach.is_coach = true";
ResultSet resultSet2a = statement.executeQuery(query2a);

System.out.println("Selection 2.a: Club name, coach name, and coach surname");
// Process the results
while (resultSet2a.next()) {
    String clubName = resultSet2a.getString( columnLabel: "name_cl");
    String coachName = resultSet2a.getString( columnLabel: "name");
    String coachSurname = resultSet2a.getString( columnLabel: "surname");
    System.out.println("Club: " + clubName);
    System.out.println("Coach: " + coachName + " " + coachSurname);
}

resultSet2a.close();
```

3. SELECT 2.b:

- Εκτελεί ένα ερώτημα για να ανακτήσει το πέναλτι, τα γκολ που σημειώθηκαν, τον χρόνο, το όνομα και το επώνυμο του παίκτη για έναν συγκεκριμένο αγώνα.

- Εκτυπώνει τα ανακτηθέντα δεδομένα.

```
// Selection 2.b
String query2b = "SELECT match_and_player_goals.penalty, match_and_player_goals.goals_s, match_and_player_goals.time, player.name, player.surname " +
        "FROM match_and_schedule " +
        "JOIN match_and_player_goals ON match_and_player_goals.id_ms = match_and_schedule.id_ms " +
        "JOIN player ON player.id_p = match_and_player_goals.id_player " +
        "WHERE match_and_schedule.id_ms = 10";
ResultSet resultSet2b = statement.executeQuery(query2b);

System.out.println("Selection 2.b: Penalty, goals scored, time, player name, and player surname");
// Process the results
while (resultSet2b.next()) {
    boolean penalty = resultSet2b.getBoolean( columnLabel: "penalty");
    boolean goalsScored = resultSet2b.getBoolean( columnLabel: "goals_s");
    Time time = resultSet2b.getTime( columnLabel: "time");
    String playerName = resultSet2b.getString( columnLabel: "name");
    String playerSurname = resultSet2b.getString( columnLabel: "surname");
    System.out.println("Penalty: " + penalty);
    System.out.println("Goals Scored: " + goalsScored);
    System.out.println("Time: " + time);
    System.out.println("Player Name: " + playerName);
    System.out.println("Player Surname: " + playerSurname);
    System.out.println("==============================");
}

resultSet2b.close();
System.out.println();
```

4 SELECT 2.c:

- Εκτελεί ένα ερώτημα για την ανάκτηση στατιστικών στοιχείων παίκτη, συμπεριλαμβανομένου του συνολικού χρόνου συμμετοχής, των τερμάτων, των πέναλτι, των κίτρινων καρτών και των κόκκινων καρτών.

- Φιλτράρει τα αποτελέσματα με βάση ένα συγκεκριμένο εύρος ημερομηνιών και το αναγνωριστικό παίκτη.

- Εκτυπώνει τα ανακτηθέντα δεδομένα.

```
// Selection 2.c
String query2c = "SELECT player.id_p, player.name, player.surname, SUM(match_and_player_time_played.time_played) AS total_time_played, player.position, " +
        "COALESCE(SUM(CASE WHEN match_and_player_goals.goals_s = true THEN 1 ELSE 0 END), 0) AS player_goals, " +
        "COALESCE(SUM(CASE WHEN match_and_player_goals.penalty = true THEN 1 ELSE 0 END), 0) AS player_penalties, " +
        "COALESCE(SUM(CASE WHEN match_and_player_cards.cards_y = true THEN 1 ELSE 0 END), 0) AS player_yellow_cards, " +
        "COALESCE(SUM(CASE WHEN match_and_player_cards.cards_r = true THEN 1 ELSE 0 END), 0) AS player_red_cards " +
        "FROM match_and_schedule " +
        "NATURAL JOIN match_and_player_time_played " +
        "JOIN player ON player.id_p = match_and_player_time_played.id_player " +
        "NATURAL JOIN match_and_player_goals " +
        "LEFT JOIN match_and_player_cards ON match_and_player_cards.id_player = match_and_player_goals.id_player " +
        "WHERE (match_and_schedule.date BETWEEN '2023-04-01' AND '2023-06-30') AND match_and_player_time_played.id_player = 14 " +
        "GROUP BY player.id_p";
ResultSet resultSet2c = statement.executeQuery(query2c);

System.out.println("Selection 2.c: Player statistics");
// Process the results
while (resultSet2c.next()) {
    int playerId = resultSet2c.getInt( columnLabel: "id_p");
    String playerName = resultSet2c.getString( columnLabel: "name");
    String playerSurname = resultSet2c.getString( columnLabel: "surname");
    Time totalPlayedTime = resultSet2c.getTime( columnLabel: "total_time_played");
    String position = resultSet2c.getString( columnLabel: "position");
    int playerGoals = resultSet2c.getInt( columnLabel: "player_goals");
    int playerPenalties = resultSet2c.getInt( columnLabel: "player_penalties");
    int playerYellowCards = resultSet2c.getInt( columnLabel: "player_yellow_cards");
    int playerRedCards = resultSet2c.getInt( columnLabel: "player_red_cards");

    System.out.println("Player ID: " + playerId);
    System.out.println("Player Name: " + playerName);
    System.out.println("Player Surname: " + playerSurname);
    System.out.println("Total Played Time: " + totalPlayedTime);
    System.out.println("Position: " + position);
    System.out.println("Player Goals: " + playerGoals);
    System.out.println("Player Penalties: " + playerPenalties);
    System.out.println("Player Yellow Cards: " + playerYellowCards);
    System.out.println("Player Red Cards: " + playerRedCards);
}
```

5. SELECT 2.d:

- Εκτελεί ένα ερώτημα για την ανάκτηση στατιστικών στοιχείων της ομάδας, συμπεριλαμβανομένων των εντός και εκτός έδρας αγώνων, των αποτελεσμάτων, των νικών, των ισοπαλιών και των ηττών.

- Φιλτράρει τα αποτελέσματα με βάση ένα συγκεκριμένο εύρος ημερομηνιών και το όνομα της ομάδας.

- Εκτυπώνει τα ανακτηθέντα δεδομένα.

```
// Selection 2.d
String query2d = "SELECT " +
        "home_matches, " +
        "away_matches, " +
        "total_home_score, " +
        "total_away_score, " +
        "total_home_score + total_away_score AS total_score, " +
        "home_matches + away_matches AS total_matches, " +
        "subquery.total_wins_home, " +
        "subquery.total_wins_away, " +
        "subquery.total_wins_home + subquery.total_wins_away AS total_wins, " +
        "subquery.total_draws_home, " +
        "subquery.total_draws_away, " +
        "subquery.total_draws_home + subquery.total_draws_away AS total_draws, " +
        "(home_matches + away_matches) - (subquery.total_wins_home + subquery.total_draws_home + subquery.total_wins_away + subquery.total_draws_away) AS total_losses, " +
        "home_matches - (subquery.total_wins_home + subquery.total_draws_home) AS total_losses_home, " +
        "away_matches - (subquery.total_wins_away + subquery.total_draws_away) AS total_losses_away " +
        "FROM ( " +
        "SELECT " +
        "SUM(CASE WHEN match_and_schedule.home_club='Aek' THEN 1 ELSE 0 END) AS home_matches, " +
        "SUM(CASE WHEN match_and_schedule.visiting_club='Aek' THEN 1 ELSE 0 END) AS away_matches, " +
        "SUM(CASE WHEN match_and_schedule.home_club='Aek' THEN match_and_schedule.home_score ELSE 0 END) AS total_home_score, " +
        "SUM(CASE WHEN match_and_schedule.visiting_club='Aek' THEN match_and_schedule.visiting_score ELSE 0 END) AS total_away_score, " +
        "SUM(CASE WHEN (match_and_schedule.home_club='Aek' AND (match_and_schedule.home_score > match_and_schedule.visiting_score)) THEN 1 ELSE 0 END) AS total_wins_home, " +
        "SUM(CASE WHEN (match_and_schedule.visiting_club='Aek' AND (match_and_schedule.visiting_score > match_and_schedule.home_score)) THEN 1 ELSE 0 END) AS total_wins_away, " +
        "SUM(CASE WHEN (match_and_schedule.home_club='Aek' AND (match_and_schedule.home_score = match_and_schedule.visiting_score)) THEN 1 ELSE 0 END) AS total_draws_home, " +
        "SUM(CASE WHEN (match_and_schedule.visiting_club='Aek' AND (match_and_schedule.visiting_score = match_and_schedule.home_score)) THEN 1 ELSE 0 END) AS total_draws_away " +
        "FROM match_and_schedule " +
        "WHERE match_and_schedule.date BETWEEN '2023-04-01' AND '2023-06-30' " +
        ") AS subquery";
ResultSet resultSet2d = statement.executeQuery(query2d);
```

```
System.out.println("Selection 2.d: Team statistics");
// Process the results
while (resultSet2d.next()) {
    int homeMatches = resultSet2d.getInt( columnLabel: "home_matches");
    int awayMatches = resultSet2d.getInt( columnLabel: "away_matches");
    int totalHomeScore = resultSet2d.getInt( columnLabel: "total_home_score");
    int totalAwayScore = resultSet2d.getInt( columnLabel: "total_away_score");
    int totalScore = resultSet2d.getInt( columnLabel: "total_score");
    int totalMatches = resultSet2d.getInt( columnLabel: "total_matches");
    int totalWinsHome = resultSet2d.getInt( columnLabel: "total_wins_home");
    int totalWinsAway = resultSet2d.getInt( columnLabel: "total_wins_away");
    int totalWins = resultSet2d.getInt( columnLabel: "total_wins");
    int totalDrawsHome = resultSet2d.getInt( columnLabel: "total_draws_home");
    int totalDrawsAway = resultSet2d.getInt( columnLabel: "total_draws_away");
    int totalDraws = resultSet2d.getInt( columnLabel: "total_draws");
    int totalLosses = resultSet2d.getInt( columnLabel: "total_losses");
    int totalLossesHome = resultSet2d.getInt( columnLabel: "total_losses_home");
    int totalLossesAway = resultSet2d.getInt( columnLabel: "total_losses_away");

    System.out.println("Home Matches: " + homeMatches);
    System.out.println("Away Matches: " + awayMatches);
    System.out.println("Total Home Score: " + totalHomeScore);
    System.out.println("Total Away Score: " + totalAwayScore);
    System.out.println("Total Score: " + totalScore);
    System.out.println("Total Matches: " + totalMatches);
    System.out.println("Total Wins (Home): " + totalWinsHome);
    System.out.println("Total Wins (Away): " + totalWinsAway);
    System.out.println("Total Wins: " + totalWins);
    System.out.println("Total Draws (Home): " + totalDrawsHome);
    System.out.println("Total Draws (Away): " + totalDrawsAway);
    System.out.println("Total Draws: " + totalDraws);
    System.out.println("Total Losses: " + totalLosses);
    System.out.println("Total Losses (Home): " + totalLossesHome);
```

7. Κλείνει τα αντικείμενα ResultSet, Statement και Connection για να απελευθερώσει πόρους. Χειρίζεται τυχόν εξαιρέσεις που εμφανίζονται κατά την εκτέλεση του κώδικα και εκτυπώνει το ίχνος στοίβας.

```
            resultSet2d.close();

            statement.close();
            connection.close();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
Main
C:\Users\Alexhs\.jdks\openjdk-18.0.1.1\bin\java.exe -Dsun.stdout.encoding=UTF-
Selection 2.a: Club name, coach name, and coach surname
Club: Olympiacos
Coach: Παναγιώτης Ρέτσος

Selection 2.b: Penalty, goals scored, time, player name, and player surname
Penalty: false
Goals Scored: true
Time: 19:59:00
Player Name: Γούσεφ
Player Surname: Ελ-Αραμπί
==============================
Penalty: false
Goals Scored: true
Time: 18:39:00
Player Name: Σεμπαστιάν
Player Surname: Παλάσιος
==============================
Penalty: false
Goals Scored: true
Time: 01:10:00
Player Name: Σεμπαστιάν
Player Surname: Παλάσιος
==============================
```

```
Selection 2.c: Player statistics
Player ID: 14
Player Name: Σεμπαστιάν
Player Surname: Παλάσιος
Total Played Time: 10:21:00
Position: Midfielder
Player Goals: 2
Player Penalties: 1
Player Yellow Cards: 0
Player Red Cards: 0

Selection 2.d: Team statistics
Home Matches: 1
Away Matches: 2
Total Home Score: 1
Total Away Score: 1
Total Score: 2
Total Matches: 3
Total Wins (Home): 1
Total Wins (Away): 1
Total Wins: 2
Total Draws (Home): 0
Total Draws (Away): 0
Total Draws: 0
Total Losses: 1
Total Losses (Home): 0
Total Losses (Away): 1
```

5 Πηγές

Για την υλοποίηση της εργασίας χρησιμοποιήσαμε τις διαφάνειες του μαθήματος και για το debugging διάφορα forums αλλά και βίντεο στο youtube