

**Εργασία στις Βάσεις Δεδομένων**  
Εφαρμογή τοπικού πρωταθλήματος ποδοσφαίρου

**ΚΑΓΙΑΦΑ ANNA**

Προπτυχιακή φοιτήτρια τμήματος ΗΜΤΥ, Πανεπιστήμιο Πατρών, up1092761@ac.upatras.gr

**ΜΕΝΔΡΙΝΟΣ ΑΝΔΡΕΑΣ**

Προπτυχιακή φοιτήτης τμήματος ΗΜΤΥ, Πανεπιστήμιο Πατρών, up1092653@ac.upatras.gr

Στην παρούσα εργασία περιγράφεται η ανάπτυξη μιας εφαρμογής τοπικού πρωταθλήματος ποδοσφαίρου, με βασικό πυρήνα τη σχεδίαση και υλοποίηση της σχετικής βάσης δεδομένων. Για την εν λόγω μοντελοποίηση αναπτύχθηκε το σχετικό μοντέλο οντοτήτων – συσχετίσεων, καθώς και το σχεσιακό σχήμα και στη συνέχεια η βάση μας υλοποιήθηκε σε SQLite. Τέλος, δημιουργήσαμε μέσω python μία διεπαφή που αποτελεί το γραφικό περιβάλλον της εφαρμογής μας. Στόχος της εφαρμογής καθίσταται η καταχώρηση δεδομένων για τα σωματεία, τους παίκτες, τα γήπεδα, καθώς και η αποτελεσματική οργάνωση των αγωνιστικών και των αγώνων, η καταγραφή των αποτελεσμάτων και η εξαγωγή χρήσιμων και αξιοσημείωτων στατιστικών για τους παίκτες, τις ομάδες και το πρωτάθλημα γενικότερα.

## 1 ΕΙΣΑΓΩΓΗ – ΠΕΡΙΓΡΑΦΗ ΜΙΚΡΟΚΟΣΜΟΥ

Για να σχεδιάσουμε τη βάση δεδομένων της εφαρμογής μας, έπρεπε σε πρώτη φάση να θέσουμε τα όρια του προβλήματός μας. Στην προσπάθειά μας, αυτή, θέλαμε ο μικρόκοσμος να αποτυπωθεί με τέτοιον τρόπο, ώστε να είναι αληθιοφανής και να ανταποκρίνεται στα δεδομένα ενός πραγματικού τοπικού πρωταθλήματος ποδοσφαίρου μιας ΕΠΣ, το οποίο καθιστά το αντικείμενο της εργασίας ενδιαφέρον, αλλά και αρκετά απαιτητικό.

Στα πλαίσια, αντά, αποφασίσαμε πως θα υπάρχει διαφορετικό πρωτάθλημα κάθε αγωνιστική περίοδο. Κάθε σεζόν, λοιπόν, θα απαρτίζεται από αγωνιστικές, σε καθεμία από τις οποίες θα διεξάγονται αγώνες μεταξύ των ομάδων που συμμετέχουν στο πρωτάθλημα και αγωνίζονται στην ίδια κατηγορία. Πιο συγκεκριμένα, διακρίνουμε τρεις κατηγορίες, Α', Β' και Γ', καθεμία από τις οποίες αποτελείται από 12 σωματεία. Αναφορικά με τα σωματεία, η ομάδα που κατεβάζει σε κάθε σεζόν στο πρωτάθλημα διαφοροποείται, καθώς μπορεί να αλλάζουν οι ποδοσφαιριστές, οι προπονητές και το γήπεδο – έδρα της. Αξίζει να αναφερθεί ότι οι ομάδες διαθέτουν ένα ρόστερ από 18 περίπου παίκτες, καθώς και έναν ή δύο προπονητές, ανάλογα με το συμβόλαιο συνεργασίας τους.

Μέσα από κατάλληλα ερωτήματα στη βάση, άλλα πιο σύνθετα, άλλα πιο απλά, καταφέρνουμε να συλλέγουμε στοιχεία για τους αγώνες και τα αποτελέσματά τους, τα γκολ, τις κάρτες και τα λεπτά συμμετοχής κάθε παίκτη, καθώς και τη βαθμολογία των ομάδων λαμβάνοντας υπόψη τις ποινές που ενδεχομένως έχουν υποστεί.

## 2 ΜΕΘΟΔΟΛΟΓΙΑ – ΒΗΜΑΤΑ ΣΧΕΔΙΑΣΜΟΥ ΤΗΣ ΒΑΣΗΣ

### 2.1 ERD

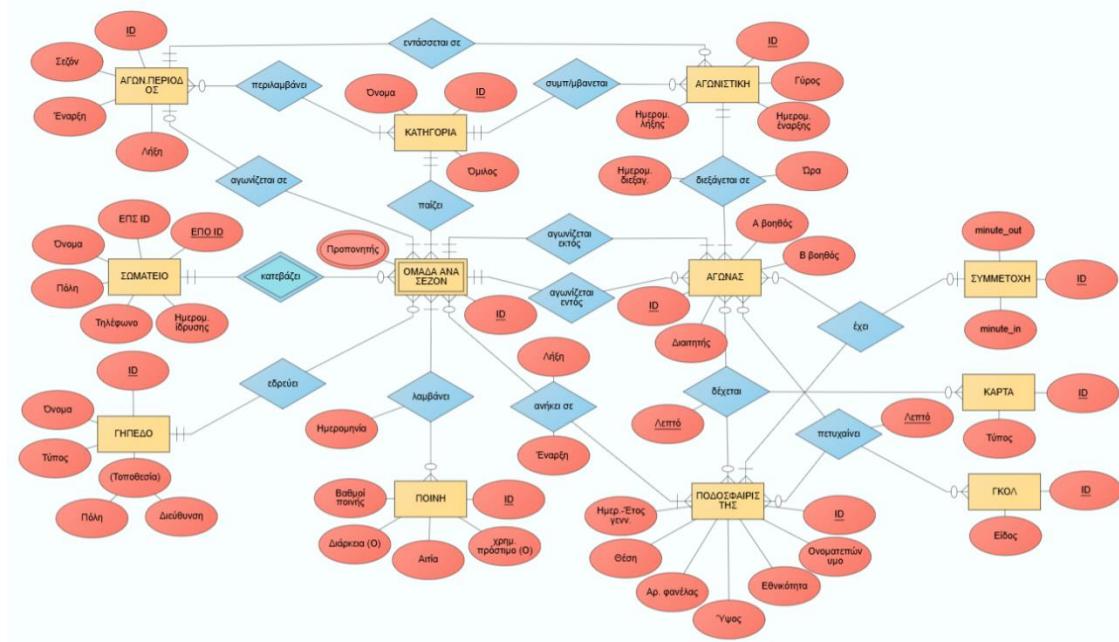
Πρώτο βήμα για τον σχεδιασμό της βάσης, όπως διδαχθήκαμε και στα πλαίσια του μαθήματος, καθίσταται δημιουργία του μοντέλου οντοτήτων – συσχετίσεων (ERD) με βάση τις απαιτήσεις του μικρόκοσμου μας.

Σε πρώτη φάση, λοιπόν, ορίσαμε τις βασικές οντότητες που κρίναμε απαραίτητο να απαρτίζουν τη βάση μας και οι οποίες είναι οι εξής: «ΑΓΩΝΙΣΤΙΚΗ ΠΕΡΙΟΔΟΣ», «ΑΓΩΝΙΣΤΙΚΗ», «ΚΑΤΗΓΟΡΙΑ», «ΣΩΜΑΤΕΙΟ», «ΟΜΑΔΑ ΑΝΑ ΣΕΖΟΝ» (διότι, όπως προαναφέρθηκε, κάθε σεζόν κάθε σωματείο λαμβάνει μέρος στο πρωτάθλημα με ομάδα με διαφορετικά χαρακτηριστικά), «ΓΗΠΕΔΟ», «ΠΟΔΟΣΦΑΙΡΙΣΤΗΣ», «ΑΓΩΝΑΣ». Έπειτα, αποφασίσαμε πως θα ήταν χρήσιμο να δημιουργήσουμε κάποιες οντότητες που μας βοηθούσαν στη συνέχεια να λαμβάνουμε πληροφορίες για τα στατιστικά των παίκτων, καθώς και για τη βαθμολογία κάθε ομάδας και άρα τη θέση τους στο πρωτάθλημα. Οι προαναφερθείσες οντότητες είναι οι εξής: «ΓΚΟΛ», «ΚΑΡΤΑ», «ΠΟΙΝΕΣ» (οι οποίες αφορούν συνολικά την ομάδα κι όχι κάθε παίκτη ξεχωριστά και δύνανται να επηρεάσουν τη βαθμολογία των ομάδων μέσω των πόντων ποινής). Μία ακόμα οντότητα που δημιουργήσαμε στη συνέχεια είναι η «ΣΥΜΜΕΤΟΧΗ», μέσω της οποίας μπορούμε να βλέπουμε αν, πόσο και ως τι (βασικός, αλλαγή) συμμετείχε ένας παίκτης σε έναν αγώνα. Τέλος, θέλαμε να φτιάξουμε μία οντότητα «ΑΠΟΤΕΛΕΣΜΑ», όπου θα καταγράφονται το σκορ των αγώνων, τα γκολ υπέρ και τα γκολ κατά, αλλά τελικά αποφασίσαμε ότι αυτά μπορούν να προκύψουν από ερωτήματα στη βάση.

Στη συνέχεια, καταγράψαμε και τοποθετήσαμε στο ERD τα γνωρίσματα κάθε οντότητας, ανάλογα με τα στοιχεία που θέλουμε να αποθηκεύουμε στη βάση μας. Κάποια από αυτά, λόγω της μοναδικότητας που θέλαμε να έχουν, τα ορίσαμε κλειδιά και τα υπογραμμίσαμε.

Το επόμενο και τελευταίο βήμα στη διαμόρφωση του ERD ήταν η δημιουργία των συσχετίσεων μεταξύ των οντοτήτων και ο καθορισμός των πληθικοτήτων, τα οποία υλοποιήθηκαν με βάση τις απαιτήσεις και τα όρια του μικρόκοσμου, όπως αυτά έχουν τεθεί παραπάνω.

Στην παρακάτω εικόνα φαίνεται, λοιπόν, το τελικό ERD που δημιουργήσαμε.



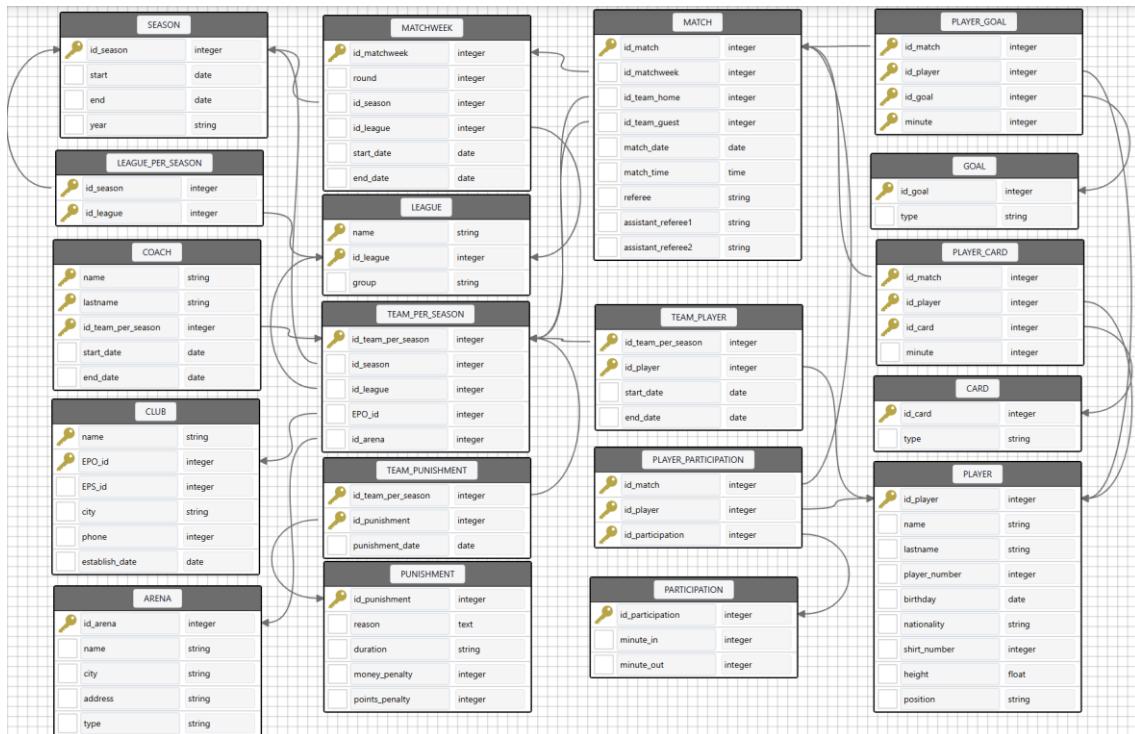
Εικόνα 1: Μοντέλο οντοτήτων – συσχετίσεων της βάσης. (<https://hci.ece.upatras.gr/erdmaker/designer>)

## 2.2 Σχεσιακό σχήμα

Το επόμενο βήμα για τον σχεδιασμό της βάσης είναι η δημιουργία του σχεσιακού σχήματος με βάση το ERD υλοποιήσαμε στο προηγούμενο βήμα. Για τον σχεδιασμό του, λοιπόν, ακολουθήθηκε η μεθοδολογία που διδαχθήκαμε στα πλαίσια του μαθήματος.

Περιληπτικά, αρχικά δημιουργήσαμε τους πίνακες για τις κύριες οντότητες με γνωρίσματα τις ιδιότητες τους και ξένα κλειδιά τα αντίστοιχα, ενώ για πλειότιμες ιδιότητες δημιουργήσαμε νέους πίνακες (Προπονητές). Νέο πίνακα δημιουργήσαμε και για την ασθενή οντότητα «ΟΜΑΔΑ ΑΝΑ ΣΕΖΟΝ» ακολουθώντας τον κανόνα για τα ξένα κλειδιά. Στη συνέχεια, δημιουργήθηκαν οι πίνακες για συσχετίσεις τύπου M:N, όπως για παράδειγμα πίνακας «ΛΑΜΒΑΝΕΙ» (μετονομάστηκε σε «TEAM\_PUNISHMENT», όπως συνέβη και με όλους τους υπόλοιπους πίνακες, για να υπάρχει συνάφεια με τους πίνακες στη βάση), ενώ τροποποιήθηκαν ανάλογα και πίνακες που δημιουργήσαμε προηγουμένως λόγω συσχετίσεων 1:N.

Συνεπώς, το σχεσιακό σχήμα που δημιουργήθηκε είναι αυτό που φαίνεται στην παρακάτω εικόνα.



Εικόνα 2: Σχεσιακό σχήμα της βάσης. (<https://150.140.186.221:1337/>)

### 3 ΜΕΘΟΔΟΛΟΓΙΑ – ΥΛΟΠΟΙΗΣΗ ΤΗΣ ΒΑΣΗΣ

#### 3.1 Υλοποίηση πινάκων

Το πρώτο βήμα για την υλοποίηση της βάσης είναι η δημιουργία των tables και των triggers, η οποία έγινε με την αξιοποίηση του εργαλείου **DB browser for sqlite**, που χρησιμοποιήσαμε και στο εργαστήριο του μαθήματος.

Σε πρώτη φάση, λοιπόν, δημιουργήσαμε τους πίνακες με την εντολή “CREATE TABLE”, στους οποίους συμπεριλάβαμε όλα τα γνωρίσματα, τα πρωτεύοντα και τα ξένα κλειδιά (με τα απαιτούμενα constraints), όπως αυτά προκύπτουν από το σχεσιακό σχήμα. Ταυτόχρονα, με CHECK ορίσαμε τις αναμενόμενες τιμές των γνωρισμάτων, ώστε να ελέγχονται τα δεδομένα που θα εισάγονται στη βάση και να ανήκουν σε συγκεκριμένα πεδία. Ένα παράδειγμα δημιουργίας πίνακα είναι το παρακάτω:

```

DROP TABLE IF EXISTS "PLAYER";
CREATE TABLE IF NOT EXISTS "PLAYER" (
    "id_player" INTEGER NOT NULL DEFAULT '0',
    "name" varchar(15) NOT NULL DEFAULT '',
    "lastname" varchar(15) NOT NULL DEFAULT '',
    "player_number" INTEGER NOT NULL DEFAULT '',
    "birthday" date NOT NULL DEFAULT '',
    "nationality" varchar(15) NOT NULL DEFAULT '',
    "shirt_number" INTEGER NOT NULL DEFAULT '' CHECK("shirt_number" BETWEEN
1 and 99),
    "height" REAL NOT NULL DEFAULT '',
    "position" varchar(15) NOT NULL DEFAULT '' CHECK("position" in
("τερματοφύλακας", "αμυντικός", "μέσος", "επιθετικός")),
    PRIMARY KEY ("id_player");
  
```

Με τον ίδιο τρόπο δημιουργήσαμε και όλους τους υπόλοιπους πίνακες, οπότε τελικά οι πίνακες που φτιάχθησαν στο DB browser for sqlite είναι συνολικά 19.

Επίσης, υλοποιήσαμε και ένα trigger με όνομα “check\_team\_categories”, ώστε όταν εισάγονται δεδομένα στη βάση σχετικά με τους αγώνες να ελέγχεται ότι αγώνες διεξάγονται μεταξύ ομάδων που αγωνίζονται στην ίδια κατηγορία. Ο κώδικας που αναπτύχθηκε για τη δημιουργία της βάσης είναι ο παρακάτω:

```
CREATE TRIGGER check_team_categories
BEFORE INSERT ON "MATCH"
FOR EACH ROW
BEGIN
SELECT RAISE(ABORT, 'Οι ομάδες δεν ανήκουν στην ίδια κατηγορία')
WHERE (SELECT l.name
      FROM (TEAM_PER_SEASON AS tp JOIN LEAGUE AS l on tp.id_league =
l.id_league)
      WHERE tp.id_team_per_season = NEW."id_team_home")
      !=
(SELECT ll.name
      FROM (TEAM_PER_SEASON AS tp1 JOIN LEAGUE AS ll on tp1.id_league =
ll.id_league)
      WHERE tp1.id_team_per_season = NEW."id_team_guest");
END;
```

Θα μπορούσαν να αναπτυχθούν κι άλλα triggers, ώστε να είμαστε σίγουροι για την ορθότητα των δεδομένων, όπως για παράδειγμα trigger που να ελέγχει ότι ένας παίκτης δέχεται κάρτα σε λεπτό που βρίσκεται στο παιχνίδι. Ωστόσο, επειδή στα πλαίσια της παρούσας εργασίας τέτοια θέματα τα έχουμε καλύψει προγραμματίζοντας με python τα δεδομένα που εισάγονται στη βάση και αφορούν πίνακες που παρουσιάζουν αρκετές συσχετίσεις, δεν υλοποιήσαμε άλλα triggers. Θα μπορούσε, βέβαια, αυτό να τεθεί ως ιδέα για ενδεχόμενη μελλοντική βελτίωση της βάσης μας.

### 3.2 Εισαγωγή δεδομένων στη βάση

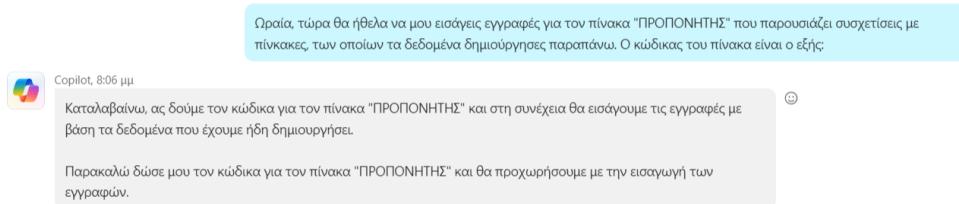
Για την εισαγωγή των δεδομένων χρησιμοποιήσαμε 2 τεχνικές:

- Χρήση ai και συγκεκριμένα του **Microsoft Copilot**
- Προγραμματισμός με **python**

Πιο συγκεκριμένα:

- Με την χρήση του Microsoft Copilot δημιουργήσαμε τα δεδομένα που εισήγαγαν στον πίνακα για τα σωματεία (CLUB), την ομάδα ανά σεζόν (TEAM\_PER\_SEASON), τα γήπεδα (ARENA) και τους προπονητές (COACH).

Στην εικόνα που ακολουθεί έχουμε ένα παράδειγμα χρήσης του Microsoft Copilot:



```

DROP TABLE IF EXISTS "ΠΡΟΠΟΝΗΤΗΣ";
CREATE TABLE IF NOT EXISTS "ΠΡΟΠΟΝΗΤΗΣ"( 
    "όνομα" varchar(15) NOT NULL DEFAULT '',
    "επώνυμο" varchar(20) NOT NULL DEFAULT '',
    "id_ομάδας_σεζόν" INTEGER NOT NULL DEFAULT '0',
    "έναρξη_συνεργασίας" date DEFAULT NULL,
    "λήξη_συνεργασίας" date DEFAULT NULL,
    "όνομα_ομάδας_σεζόν" varchar(20) NOT NULL DEFAULT '',
    "ΕΠΣ_id" INTEGER NOT NULL DEFAULT '0',
    "ΕΠΟ_id" INTEGER NOT NULL DEFAULT '0',
    CONSTRAINT "προπονητής_id_ομ" FOREIGN KEY("id_ομάδας_σεζόν") REFERENCES "ΟΜΑΔΑ ΑΝΑ ΣΕΖΟΝ"
    ("id_ομάδας_σεζόν") ON DELETE SET DEFAULT NULL ON UPDATE CASCADE,
    CONSTRAINT "προπονητής_όνομα_ομ" FOREIGN KEY("όνομα_ομάδας_σεζόν") REFERENCES "ΟΜΑΔΑ ΑΝΑ ΣΕΖΟΝ"
    ("όνομα") ON UPDATE CASCADE,
    CONSTRAINT "προπονητής_ΕΠΣ_id" FOREIGN KEY("ΕΠΣ_id") REFERENCES "ΟΜΑΔΑ ΑΝΑ ΣΕΖΟΝ"("ΕΠΣ_id") ON
    UPDATE CASCADE,
    CONSTRAINT "προπονητής_ΕΠΟ_id" FOREIGN KEY("ΕΠΟ_id") REFERENCES "ΟΜΑΔΑ ΑΝΑ ΣΕΖΟΝ"("ΕΠΟ_id") ON
    UPDATE CASCADE,
    PRIMARY KEY("όνομα", "επώνυμο", "id_ομάδας_σεζόν", "όνομα_ομάδας_σεζόν", "ΕΠΣ_id", "ΕΠΟ_id")
);

```

```

INSERT INTO "ΠΡΟΠΟΝΗΤΗΣ" ("όνομα", "επώνυμο", "id_ομάδας_σεζόν", "έναρξη_συνεργασίας",
"λήξη_συνεργασίας", "όνομα_ομάδας_σεζόν", "ΕΠΣ_id", "ΕΠΟ_id") VALUES
-- Κατηγορία Α
('Γιάννης', 'Παπαδόπουλος', 1, '2023-01-01', '2023-12-31', 'Παναχαική', 1, 101),
('Κώστας', 'Νικολάου', 2, '2023-01-01', '2023-12-31', 'ΑΕΚ Πατρών', 1, 102),
('Δημήτρης', 'Σταματόπουλος', 3, '2023-01-01', '2023-12-31', 'Ατρόμιτος Πατρών', 1, 103),
('Πέτρος', 'Αναστασίου', 4, '2023-01-01', '2023-12-31', 'Θύελλα Πατρών', 1, 104),
('Αλέξανδρος', 'Μακρής', 5, '2023-01-01', '2023-12-31', 'Δέξα Παραλίας', 1, 105),
('Σπύρος', 'Κωνσταντίνου', 6, '2023-01-01', '2023-12-31', 'Αναγέννηση Πατρών', 1, 106),
('Μιχάλης', 'Παπαδημητρίου', 7, '2023-01-01', '2023-12-31', 'Απόλλων Εγλυκάδας', 1, 107),
('Νίκος', 'Αθανασίου', 8, '2023-01-01', '2023-12-31', 'ΑΕ Αρόης', 1, 108),
('Γιώργος', 'Παναγιωτόπουλος', 9, '2023-01-01', '2023-12-31', 'Πανμοβιακός Ριόλου', 1,
109),
('Χρήστος', 'Διαμαντόπουλος', 10, '2023-01-01', '2023-12-31', 'Αχαϊκή', 1, 110),
('Ανδρέας', 'Καραγιάννης', 11, '2023-01-01', '2023-12-31', 'Διαγόρας Βραχνείκων', 1,
111),
('Θανάσης', 'Παπαδόπουλος', 12, '2023-01-01', '2023-12-31', 'Αστέρας Τσουκαλείκων', 1,
112),

```

Εικόνα 3: Εισαγωγή δεδομένων στον πίνακα COACH με το Microsoft Copilot

Επιπρόσθετα, με χρήση του Microsoft Copilot ήταν εύκολη η εισαγωγή δεδομένων σε πίνακες με λίγες εγγραφές, όπως στους “SEASON”, “LEAGUE”, “LEAGUE\_PER\_SEASON”. Ενώ τους πίνακες “PUNISHMENT” και “TEAM\_PUNISHMENT” που αναφέρουν ποινές, όπως αφαίρεσης βαθμών από την τρέχουσα σεζόν, τους δημιουργήσαμε τυχαία με το χέρι.

➤ Τους πίνακες που απαιτούσαν περισσότερες εγγραφές τους υλοποιήσαμε με συναρτήσεις στην python.

Αυτό που κάναμε, αρχικά, είναι να βρούμε έναν τρόπο να «επικοινωνύμε» με την βάση μας. Να μπορούμε, δηλαδή, να διαβάζουμε στοιχεία από τους ήδη υπάρχοντες πίνακες και να γράφουμε πίσω στην βάση. Για να το πετύχουμε, αντό, κάναμε **import** την βιβλιοθήκη **sqlite3** της python. Με τις εντολές που μας παρείχε αυτή η βιβλιοθήκη δημιουργήσαμε τις συναρτήσεις: `def readSQL(dbfilename, sql, data = None):`

```

db = sqlite3.connect(dbfilename)
cur = db.cursor()
if data == None:
    cur.execute(sql)
else:
    cur.execute(sql, data)
result = cur.fetchall()
db.close()
return result

def writeSQL(dbfilename, sql, data = None):
    db = sqlite3.connect(dbfilename)
    cur = db.cursor()

```

```

        if data == None:
            cur.execute(sql)
        else:
            cur.execute(sql, data)
            # result = cur.fetchall()
            db.commit()
            db.close()
        return 1
    
```

Με την **readSQL** αντλούμε ήδη υπάρχοντα στοιχεία από την βάση μας και με την **writeSQL** εισάγουμε δεδομένα στους πίνακες.

Στην συνέχεια, αφού βρήκαμε τρόπο να αλληλεπιδρούμε με την βάση μας, έπρεπε να εισάγουμε τα δεδομένα. Η σειρά με την οποία «γεμίζουμε» τους πίνακες, παίζει πολύ σημαντικό ρόλο διότι πολλά γνωρίσματα μερικών πινάκων συσχετίζονται με άλλα των υπόλοιπων, καθώς περιέχουν ξένα κλειδιά. Για παράδειγμα, δεν γίνεται να δημιουργήσουμε δεδομένα για τον πίνακα “PLAYER\_GOAL” χωρίς να έχουμε για τον πίνακα “PLAYER”.

Για αυτόν τον λόγο, οι πρώτοι πίνακες για τους οποίους έπρεπε να φτιάξουμε δεδομένα ήταν οι “PLAYER” και “TEAM\_PLAYER”. Για να βρούμε **name** και **lastname** των ποδοσφαιριστών δημιουργήσαμε 2 μήτρες, οι οποίες περιείχαν μια πληθώρα από ελληνικά ονόματα και επίθετα. Πιο συγκεκριμένα :

```

first = [
    'Γώργος', 'Ηλίας', 'Αποστόλης', 'Δημήτρης', 'Λευτέρης', 'Φάνης', 'Ξενοφών',
    'Λεωνίδας', 'Λέων', 'Μάριος', 'Νίκος', 'Τάκης', 'Πίπος', 'Μάνος', 'Νέτρος',
    'Κωνσταντίνος', 'Ιωνάς', 'Ανδρέας', 'Προκόπης', 'Γιάννης', 'Χρόνης', 'Θανάσης',
    'Χαρούλης', 'Τρίψωνας', 'Σπύρος', 'Παύλος', 'Γεωργίη', 'Σαμουῆλ',
    'Τζόρτζης', 'Τρίψωνας', 'Άδημπης', 'Τόνη', 'Μίκης', 'Σάκης', 'Γεώργιος', 'Κώστας', 'Πάρις',
    'Τζίμης', 'Φαίδωνας', 'Λάμπρος', 'Στάθης', 'Φίλιππος', 'Άρης', 'Αριστείδης', 'Ημηρος',
    'Οδυσσέας', 'Θέμης', 'Θεοφιλοκλής', 'Περικλής', 'Αλκης', 'Ιάσοςωνας', 'Ορφέας', 'Δημοθένης',
    'Μαρίνος', 'Θάνος', 'Μανούσος', 'Μάρκελος', 'Μίλτος', 'Πρίσαρος', 'Ικαρος', 'Νότης', 'Άγης',
    'Προκόπης', 'Ζήνωνας', 'Σώτης', 'Στέλιος', 'Λουκάς', 'Παντελής', 'Νεκτάριος', 'Άκης',
    'Ζάχος', 'Ζαχαρίας', 'Ηρακλής', 'Ιούλιος', 'Στράτης', 'Στρατής', 'Παυσανίας', 'Ανέστης',
    'Μάρκος', 'Πλάτωνας', 'Άγγελος', 'Γιάννης', 'Κώστας', 'Δημήτρης', 'Αλέξανδρος', 'Παναγιώτης',
    'Ανδρέας', 'Στέφανος', 'Αναστάτιος', 'Βασίλειος', 'Χαράλαμπος', 'Εμμανουήλ', 'Συπορίδην',
    'Ιωάννης', 'Πέτρος', 'Γιάννης', 'Γιάννης', 'Γιάννης', 'Γιώργος', 'Γιώργος', 'Γιώργος',
    'Δημήτρης', 'Δημήτρης', 'Δημήτρης', 'Ανδρέας', 'Κώστας', 'Κώστας'
];

last = [
    'Γιατζόγηλου', 'Δημητρίου', 'Ιωάννου', 'Παποπέτρου', 'Σταματίου', 'Αράπογλου', 'Πελεχιθάνογλου',
    'Αποστόλου', 'Βασιλείου', 'Κυπριανού', 'Χριστοδούλου', 'Τσακάλω', 'Σαρδαργλου', 'Άγγελογλου',
    'Χατζηπανάνου', 'Χατζηδημητρίου', 'Χατζηκριστοδούλου', 'Χατζηπανατόλου', 'Κόντογλου', 'Εμμανουήλ',
    'Χατζηβασιλείου', 'Παπασταμάτου', 'Παπαϊωάννου', 'Παπαδημητρίου', 'Σήμουρ', 'Παπαστύρου',
    'Παπαϊωστάλου', 'Παπαβασιλείου', 'Κανταντίνου', 'Παπακωνσταντίνου', 'Χατζηπούρου',
    'Αθανασίου', 'Παπαθανασίου', 'Αλεξίου', 'Χατζηλεξίου', 'Ευσταθίου', 'Καλύνου', 'Μάρκου',
    'Χατζηπευτσαΐου', 'Πρωκοπίου', 'Σωτηρίου', 'Παπαωτηρίου', 'Ντυνόν', 'Χατζημάρκου', 'Βησσέου',
    'Πολυχρονίου', 'Στεργίου', 'Σακελλαρίου', 'Θωτίου', 'Χέτζη', 'Ραφηνή', 'Πλάτωνας',
    'Οὐκονόνου', 'Παπαϊωνάνου', 'Ελευθερίου', 'Χατζηλευθερίου', 'Σιμεών', 'Στάνκογλου',
    'Ιωάνη', 'Μιχαήλ', 'Ιωακείμ', 'Ναθαναήλ', 'Ματθίου', 'Χατζημαθίου', 'Παπαδόπουλος',
    'Σίγμα', 'Κόππας', 'Γεωργίου', 'Χατζηγεωργίου', 'Παπαγεωργίου', 'Χατζηνικαλάου',
    'Παπαζηκαλή', 'Γεωργίανου', 'Καραμανώφ', 'Χατζηλιακίου', 'Τριανταφύλλου', 'Γρηγορίου', 'Χατζηφωτίου',
    'Καρανευρίου', 'Στηρίου', 'Αντωνίου', 'Παπαχρήστου', 'Ανημητρίου', 'Παπαδόπουλος', 'Νικολάου', 'Σταματόπουλος',
    'Αναστασίου', 'Μακρής', 'Κανταντίνου', 'Παπαδημητρίου', 'Αθανασίου', 'Παπακωντάπουλος', 'Διαμαντόπουλος',
    'Καραγιάννης', 'Παπαδόπουλος', 'Μακρής', 'Παπαδόπουλος', 'Ιωάννου', 'Δημητρίου', 'Μιχαηλίδης', 'Αναγνωστάπουλος',
    'Παπακωνσταντίνου', 'Γεωργίου', 'Κυριακίδης', 'Αλεξίου', 'Παπαδόπουλος', 'Αναστασίου', 'Μακρής',
    'Κανταντίνου', 'Παπαδημητρίου', 'Παπαδόπουλος', 'Θεονασίου', 'Παναγιωτάπουλος', 'Διαμαντάπουλος', 'Καραγιάννης',
    'Παπαδόπουλος', 'Ιωάννου', 'Δημητρίου', 'Μιχαηλίδης', 'Αναγνωστάπουλος', 'Μενέρνας', 'Καγιάδας',
    'Κωσταντέλης', 'Λιάθης', 'Μπουζουκής', 'Χατζηγιανένης', 'Τζόλης', 'Καϊσαρης', 'Ιωάννου', 'Οικονόμου', 'Ευαγγέλου'
];
    
```

Έτσι για κάθε παίκτη, επιλέγουμε τυχαία ένα **όνομα** και ένα **επίθετο** από τους πίνακες first και last. Για την τυχαία επιλογή έχουμε δημιουργήσει την συνάρτηση **RandItem()**.

```

def randItem(list):
    return list[ random.randint(0, len(list)-1) ]
    
```

Για το γνώρισμα **birthday**, υλοποιήσαμε μια συνάρτηση **randDate()** η οποία μας επέστρεφε μια ημερομηνία γέννησης ανάμεσα στο 1992 και το 2006. Συγκεκριμένα :

```

def randDate(): # yyyy-mm-dd
    y = str(random.randint(1992, 2006))
    m = random.randint(1,12)
    if m<10:
        ms = '0'+ str(m)
    else:
        
```

```

    ms = str(m)

    d = random.randint(1,28)

    if d<10:

        ds = '0'+ str(d)

    else:

        ds = str(d)

    return y + '-' + ms + '-' + ds

```

Για το γνώρισμα **nationality** (εθνικότητα), επιλέγουμε πάλι μια τυχαία επιλογή από τον πίνακα:

```

nationality = [
    'Ελλάδα', 'Ελλάδα', 'Ελλάδα',
    'Σερβία', 'Ελλάδα',
    'Κύπρος', 'Αλβανία', 'Ελλάδα',
    'Ελλάδα', 'Ελλάδα', 'Κύπρος',
    'Νιγηρία', 'Ελλάδα'
]

```

Επειδή, όπως είναι λογικό η πλειοψηφία των παικτών είναι Έλληνες, δίνουμε αρκετές φορές την εθνικότητα 'Έλλαδα' για να είναι και πιο πιθανό να την επιλέξουμε.

Για το γνώρισμα **height** επιλέγουμε για κάθε έναν αριθμό ανάμεσα στο 1.45 και το 2.05.

Για το **position** και το **shirt\_number** ακολουθούμε την εξής λογική. Για τους πρώτους 15 ποδοσφαιριστές μια ομάδας, η θέση και ο αριθμός της φανέλας τους σχετίζονται. Για παράδειγμα ο αριθμός 1 και 15 μπορεί να καταχωρηθεί μόνο σε τερματοφύλακες ενώ το 2 μόνο σε αμυντικούς. Οι παίκτες από τον 16<sup>ο</sup> και μετά μπορεί να αγωνίζονται σε οποιαδήποτε θέση. Αυτό φαίνεται στο παρακάτω πίνακα :

```

shirt = [
    ['01', "τερματοφύλακας"], ['02', "αμυντικός"], ['03', "αμυντικός"],
    ['04', "αμυντικός"], ['05', "αμυντικός"], ['06', "μέσος"], ['07', "μέσος"],
    ['08', "μέσος"], ['09', "μέσος"], ['10', "επιθετικός"], ['11', "επιθετικός"],
    ['12', "αμυντικός"], ['13', "μέσος"], ['14', "επιθετικός"], ['15', "τερματοφύλακας"],
    ['16', ""], ['17', ""], ['18', ""], ['19', ""], ['20', ""], ['21', ""], ['22', ""],
    ['23', ""], ['24', ""],
]

```

Από τον αριθμό 16 και μετά, η τυχαία θέση υπολογίζεται από τον πίνακα :

```

position = [
    "αμυντικός", "αμυντικός",
    "μέσος", "μέσος",
    "επιθετικός"
]

```

Τέλος, κάθε παίκτης έχει ένα μοναδικό κλειδί, το **id\_player** το οποίο εξαρτάται από την ομάδα στην οποία αγωνίζεται.

```

"id_player": (team_id+1) * 20 + index,

```

Για παράδειγμα οι παίκτες της πρώτης ομάδας θα μπορούν να έχουν id από 40 μέχρι 59, της δεύτερης από 60 μέχρι 79 κτλ.

Συνολικά, όλα αυτά τα γνωρίσματα υπολογίζονται στην συνάρτηση **Player** η οποία παίρνει ως ορίσματα το **id** της ομάδας για την οποία δημιουργούμε τους παίκτες και το **index** από το οποίο δημιουργούμε τα **id** και τα **shirt\_numbers**.

Συνάρτηση για την δημιουργία παίκτη :

```

def player(index, team_id):

    name = randItem(first)

    lastname = randItem(last)

```

```

birthday = randDate()

height = str(random.randint(145,204)/100)

pos = shirt[index][1]

if pos == "":
    pos = randItem(position)

return {

    "id_player": (team_id+1) * 20 + index,
    "name": name,
    "lastname": lastname,
    "birthday": birthday,
    "nationality": randItem(nationality),
    "shirt_number": shirt[index][0],
    "height": height,
    "position": pos
}

```

Για να δημιουργήσουμε, όμως, παίκτες πρέπει να γνωρίζουμε και την ομάδα στην οποία αγωνίζονται. Για αυτό, πρώτα πρέπει να διαβάσουμε και να αντλήσουμε από την βάση μας τα δεδομένα για τις ομάδες που αγωνίζονται κάθε σεζόν. Για αυτό θα χρησιμοποιήσουμε την συνάρτηση **readSQL** που εξηγήσαμε παραπάνω. Συγκεκριμένα :

```
database = 'benglish.db'
```

```

teamsSql = readSQL(database,'''
SELECT
    tps.id_team_per_season, tps.EPO_id,
    c.name, c.city,
    s.start, s.end
FROM TEAM_PER_SEASON AS tps
LEFT JOIN CLUB c ON c.EPO_id = tps.EPO_id
LEFT JOIN SEASON s ON s.id_season = tps.id_season
'''')

```

Συνεπώς αν πάρουμε το 1<sup>o</sup> στοιχείο του πίνακα **teamsSql** παίρνουμε τα **id\_per\_season**, αν πάρουμε το 5<sup>o</sup> έχουμε την ημερομηνία έναρξης της season κτλ. Θεωρούμε ότι τα συμβόλαια των παικτών ξεκινάνε με την αρχή της σεζόν και ολοκληρώνονται στο τέλος της.

Με όλα αυτά τα στοιχεία που έχουμε συλλέξει μπορούμε να δώσουμε πλήρη δεδομένα στο πίνακες **PLAYER** και **TEAM\_PLAYER**. Όπως προαναφέραμε, θεωρούμε ότι έχουμε 36 ομάδες με 18 ποδοσφαιριστές η κάθε ομάδα. Συνολικά λοιπόν η βάση μας έχει  $18 \times 36 = 648$  παίκτες.

```
# setup
number_of_teams = 36
number_of_players = 18
database = 'benglish.db'
```

Τα δεδομένα τα αντλούμε με την συνάρτηση **create\_team\_players()**. Με την συνάρτηση **writeSQL** εισάγουμε τα δεδομένα.

```

def create_team_players(): # fill PLAYER and TEAM_PLAYER tables
    for ti in range(0, number_of_teams): # from ARENA

```

```

players = []

for i in range(0, number_of_players):
    plr = player(i, teamsSql[ti][0])
    writeSQL(database, '''
        INSERT INTO PLAYER (
            "id_player",
            "name",
            "lastname",
            "birthday",
            "nationality",
            "shirt_number",
            "height",
            "position"
        ) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
    ''', (
        plr["id_player"],
        plr["name"],
        plr["lastname"],
        plr["birthday"],
        plr["nationality"],
        plr["shirt_number"],
        plr["height"],
        plr["position"]
    ))
    players.append(plr);

clubs.append({
    "id_team_per_season": teamsSql[ti][0],
    "name": teamsSql[ti][2],
    "players": players,
    "start_date": teamsSql[ti][4],
    "end_date": teamsSql[ti][5],
})

for p in players:
    writeSQL(database, '''
        INSERT INTO TEAM_PLAYER (
            id_team_per_season,
            id_player,
            start_date,
            end_date
    ''')

```

```

        ) VALUES (?, ?, ?, ?, ?)

        ''', (
            teamsSql[ti][0],
            p["id_player"],
            teamsSql[ti][4],
            teamsSql[ti][5]
        )
    )

```

Μερικές από τις πλειάδες που δημιουργούμε στην βάση μας :

**PLAYER**

	id_player	name	lastname	birthday	nationality	shirt_number	height	position
1	40	Ξενοφών	Σίγυα	1993-07-27	Ελλήση	1	1.54	τερματοφύλακας
2	41	Ζάχος	Αράπογλου	1993-06-25	Ελλήση	2	1.89	ομοντικός
3	42	Κωνσταντίνος	Χατζημάνου	1995-07-06	Ελλήση	3	1.74	ομοντικός
4	43	Πρίαμος	Ιανόου	2002-12-14	Νιγηρία	4	1.93	ομοντικός
5	44	Νεκτάριος	Σταματόπουλος	1992-10-14	Ελλήση	5	1.86	ομοντικός
6	45	Ζάχος	Χατζητσιάθη	1995-01-18	Ελλήση	6	1.52	μέσος
7	46	Μανούσος	Κωνσταντίνου	2000-06-27	Ελλήση	7	1.74	μέσος
8	47	Μίκης	Μακρής	2003-01-27	Αλβανία	8	1.75	μέσος
9	48	Θέμης	Συμεών	1998-06-26	Ελλήση	9	1.61	μέσος
10	49	Ηλίας	Σταματόπουλος	2004-07-03	Ελλήση	10	2.01	επιθετικός

**team\_per\_season**

	id_team_per_season	id_player	start_date	end_date
1	1	40	2023-09-01	2024-05-31
2	1	41	2023-09-01	2024-05-31
3	1	42	2023-09-01	2024-05-31
4	1	43	2023-09-01	2024-05-31
5	1	44	2023-09-01	2024-05-31
6	1	45	2023-09-01	2024-05-31
7	1	46	2023-09-01	2024-05-31
8	1	47	2023-09-01	2024-05-31
9	1	48	2023-09-01	2024-05-31
10	1	49	2023-09-01	2024-05-31

Στην συνέχεια, έπρεπε να φτιάξουμε τα δεδομένα για τους πίνακες **MATCHWEEK** και **MATCH**. Κάθε κατηγορία διαθέτει 12 ομάδες οπότε είναι σημαντικό να φτιάξουμε το πρόγραμμα αγώνων για όλες αυτές τις ομάδες έτσι ώστε στο τέλος του πρωταθλήματος όλες να έχουν παίξει με όλες 2 φορές (μια εντός και μία εκτός). Για αυτό δημιουργήσαμε μια συνάρτηση **create\_round\_robin\_schedule** η οποία παίρνει ως άριστα κάποιες ομάδες και δημιουργεί το προαναφερθέν. Για παράδειγμα, στην πρώτη αγωνιστική η ομάδα 1 παίζει με την ομάδα 12, η 2 με την 11 κτλ. Ο αλγόριθμος που υπολογίζει το πρόγραμμα αγώνων φαίνεται παρακάτω.

```

def create_round_robin_schedule(teams):
    """
    Δημιουργεί πρόγραμμα πρωταθλήματος με γύρους όπου κάθε αγωνιστική
    έχει τον μέγιστο αριθμό αγώνων και καμία ομάδα δεν παίζει πάνω από μία φορά.

    """

    if len(teams) % 2 != 0:
        teams.append("Ρεπό") # Προσθήκη "Ρεπό" αν ο αριθμός των ομάδων είναι περιττός

    num_teams = len(teams)
    schedule = []
    for round_num in range(num_teams - 1):
        round_matches = [

```

```

for i in range(num_teams // 2):
    match = (teams[i], teams[num_teams - 1 - i])
    round_matches.append(match)
schedule.append(round_matches)

# Περιστροφή των ομάδων για τον επόμενο γύρο
teams = [teams[0]] + [teams[-1]] + teams[1:-1]

return schedule

```

Με αυτήν την συνάρτηση υπολογίζουμε τους 11 αγώνες που θα δώσει η κάθε ομάδα. Οι υπόλοιποι 11 αγώνες τις αγωνιστικής περιόδου είναι οι ίδιοι, απλά το μόνο που αλλάζει είναι η γηπεδούχος και η φιλοξενούμενη ομάδα. Το πλήρες πρόγραμμα υπολογίζεται στην συνάρτηση **schedule\_champ**:

```

def create_round_robin_schedule(teams):
    if len(teams) % 2 != 0:
        teams.append("Ρεπό") # Προσθήκη "Ρεπό" αν ο αριθμός των ομάδων είναι περιττός
    num_teams = len(teams)
    schedule = []
    for round_num in range(num_teams - 1):
        round_matches = []
        for i in range(num_teams // 2):
            match = (teams[i], teams[num_teams - 1 - i])
            round_matches.append(match)
        schedule.append(round_matches)
        # Περιστροφή των ομάδων για τον επόμενο γύρο
        teams = [teams[0]] + [teams[-1]] + teams[1:-1]

    return schedule

```

Επιπλέον, όλοι οι αγώνες πραγματοποιούνται μια συγκεκριμένη ημερομηνία. Η μια αγωνιστική απέχει από την άλλη μια εβδομάδα για αυτό θεωρούμε ότι κάθε ομάδα παίζει ανά 7 ημέρες. Για να δημιουργήσουμε τις τυχαίες ημερομηνίες χρησιμοποιούμε τις συναρτήσεις:

```

import datetime

def days_after(base, n):
    base_date = datetime.datetime.strptime(base, "%Y-%m-%d")
    end_date = base_date + datetime.timedelta(days=n)
    #print(base, end_date)
    return end_date.strftime("%Y-%m-%d")

```

```

def matchdays_of_round(base, round):
    return (days_after(base, round*7), days_after(base, round*7 +3))

```

Επομένως, έχοντας συλλέξει όλα αυτά μπορούμε να δημιουργήσουμε δεδομένα για τους δύο πίνακες.

Η συνάρτηση που γράφει δεδομένα στους πίνακες είναι η :

```

def create_schedule(season_id, league_id):
    writeSQL(database, '''

```

```

        DELETE FROM MATCHWEEK WHERE id_season = ? AND id_league = ?
        ''', (season_id, league_id)
    )
    min_mw = (season_id * 10 + league_id) * 100 -1
    max_mw = (season_id * 10 + league_id) * 100 + 99
    writeSQL(database, '''

        DELETE FROM MATCH WHERE id_matchweek > ? AND id_matchweek < ?
        ''', (min_mw, max_mw)
    )
    matchweeks = schedule_champ(champ)
    round = 0
    for mw in matchweeks:
        round = round + 1
        id_matchweek = (season_id * 10 + league_id) * 100 + round
        weekdates = matchdays_of_round(seas[season_id][1], round)
        writeSQL(database, '''

            INSERT INTO MATCHWEEK(
                id_matchweek, round, id_season, id_league,
                start_date, end_date
            ) VALUES (?, ?, ?, ?, ?, ?, ?)
            ''', (
                id_matchweek,
                round,
                season_id,
                league_id,
                weekdates[0],
                weekdates[1]
            ))
        gmi = 0
        for gm in mw:
            gmi = gmi +1
            ref0 = randItem(first)+ " "+ randItem(last);
            ref1 = randItem(first)+ " "+ randItem(last);
            ref2 = randItem(first)+ " "+ randItem(last);
            id_match = id_matchweek * 10 + gmi
            writeSQL(database, '''

                INSERT INTO MATCH (id_match, id_matchweek, match_date, match_time,
                id_team_home, id_team_guest,
                referee, assistant_referee1, assistant_referee2 )
            ''')

```

```

VALUES  (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
'',
    id_match,
    id_matchweek,
    weekdates[0],
    '16:00',
    gm[0], gm[1],
    ref0, ref1, ref2
)

```

Τη συνάρτηση, αυτήν, την εκτελούμε 3 φορές (1 για κάθε league (κατηγορία) με τις αντίστοιχες ομάδες που συμμετέχουν στην κατηγορία (πίνακας **champ**)) και πάρνουμε στο σύνολο :

22 (γύρους) x 3 (κατηγορίες) = 66 εγγραφές στον πίνακα **MATCHWEEK**

22 x 3 x 6 (αγώνες / αγωνιστική) = 396 εγγραφές για τον πίνακα **MATCH**

Επίσης, τα ονόματα των διαιτητών συλλέγονται με όμοιο τρόπο με αυτά των ποδοσφαιριστών.

Σημείωση για τα **id\_match** και **id\_matchweek**:

- **id\_matchweek** είναι ένας 4-ψήφιος αριθμός που περιέχει το **id\_season** σαν πρώτο ψηφίο, το **id\_league** σαν δεύτερο ψηφίο και τον αριθμό της αγωνιστικής στα τελευταία 2 ψηφία.
- **id\_match** είναι ένας 5-ψήφιος αριθμός που περιέχει το **id\_matchweek** στα πρώτα 4 ψηφία και τον αριθμό του αγώνα στο 5<sup>o</sup> ψηφίο.

Για παράδειγμα για το match με **id\_match** → 12083 καταλαβαίνουμε ότι είναι ο 3<sup>ος</sup> αγώνας της 1<sup>ης</sup> σεζόν, της 2<sup>ης</sup> κατηγορίας, της 8<sup>ης</sup> αγωνιστικής.

Μερικά ενδεικτικά στοιχεία από τους πίνακες **MATCHWEEK KAI MATCH** :

Table: MATCH								
	<b>id_match</b>	<b>id_matchweek</b>	<b>match_date</b>	<b>match_time</b>	<b>id_team_home</b>	<b>id_team_guest</b>	<b>referee</b>	<b>assistant_referee1</b>
1	11011	1101	2023-09-08	16:00	1	12 Επυρίδων Χατζηιωάννου	Σπυρίδων Μάρκου	Μάριος Κωσταντέλιας
2	11012	1101	2023-09-08	16:00	2	11 Γιάννης Πολυχρονίου	Δημήτρης Αλεξίου	Τζίμης Ιωσήφ
3	11013	1101	2023-09-08	16:00	3	10 Δημήτρης Παπαδημητρίου	Δημοσθένης Παπακωνσταντίνου	Στρατής Κόντογλου
4	11014	1101	2023-09-08	16:00	4	9 Στράτος Αλεξίου	Μανούσος Στεργίου	Τζώρτης Αναστασίου
5	11015	1101	2023-09-08	16:00	5	8 Γεώργιος Παπαπέτρου	Αποστόλης Αλεξίου	Μηάμης Παπακωνσταντίνου
6	11016	1101	2023-09-08	16:00	6	7 Πρίαμος Τριανταφύλλου	Μανούσος Παπαποστόλου	Αριστείδης Γεωργίου
7	11021	1102	2023-09-15	16:00	1	11 Μηάμης Εμμανουήλ	Τάκης Χατζηγεωργίου	Θεμιστοκλής Καραγιάννης

Table: MATCHWEEK						
	<b>id_matchweek</b>	<b>round</b>	<b>id_season</b>	<b>id_league</b>	<b>start_date</b>	<b>end_date</b>
1	1101	1	1	1	2023-09-08	2023-09-11
2	1102	2	1	1	2023-09-15	2023-09-18
3	1103	3	1	1	2023-09-22	2023-09-25
4	1104	4	1	1	2023-09-29	2023-10-02

Στην συνέχεια, έχοντας τους αγώνες, έπρεπε να βρούμε δεδομένα για τις κάρτες, τα γκολ και τη συμμετοχή των παικτών σε αυτά, δηλαδή για τους πίνακες **GOAL**, **PLAYER\_GOAL**, **CARD**, **PLAYER\_CARD**, **PARTICIPATION**, **PLAYER\_PARTICIPATION**.

Για τους πίνακες **GOAL** και **CARD**:

```

def create_types():

    # card

```

```

writeSQL(database, '''

    INSERT INTO CARD (id_card, type)
    VALUES (?, ?), (?, ?);

'', (1, "κίτρινη", 2, "κόκκινη")

)

# goal types

writeSQL(database, '''

    INSERT INTO GOAL (id_goal, type)
    VALUES (?, ?), (?, ?);

'', (1, "γκολ_υπέρ", 2, "αυτογκόλ")

)

```

Το μόνο που κοιτάμε είναι το γκολ αν είναι υπέρ ή αυτογκόλ και το χρώμα της κάρτας.

**Συμβιβασμοί για τη δημιουργία δεδομένων :**

- **Όσον αφορά τη συμμετοχή των παικτών :**

Σε κάθε αγώνα, κάθε ομάδα αγωνίζεται με 11 παίκτες και έχει δικαίωμα για 3 αλλαγές, αλλά δεν είναι υποχρεωμένη να τις κάνει όλες. Επίσης, πάντα πρέπει να βρίσκεται στον αγωνιστικό χώρο ένας τερματοφύλακας. Επομένως, οι 10 από τους 11 βασικούς προκύπτουν από τους παίκτες που είναι αμυντικοί, επιθετικοί και μέσοι.

- **Όσον αφορά τις κάρτες των παικτών :**

Κάθε ομάδα μπορεί να δεχτεί μέχρι 5 κίτρινες κάρτες σε έναν αγώνα ενώ έχει 10% πιθανότητα να δεχτεί κόκκινη κάρτα. Οι κάρτες δίνονται σε τυχαίο λεπτό, στο οποίο ο συγκεκριμένος παίκτης αγωνίζεται. Επίσης, όταν ένας παίκτης δεχθεί κόκκινη κάρτα δεν μπορεί να παίξει στον επόμενο αγώνα της ομάδας του.

- **Όσον αφορά τα γκολ των παικτών :**

Κάθε γκολ έχει 10% πιθανότητα να είναι αυτογκόλ και 90% κανονικό γκολ.

```

max_noch = 3      # number of changes
basikoi = 10
max_yellow = 5
red_pc = 10       # red-card percentage
owngoal_pc = 10

```

Για την δημιουργία δεδομένων για συμμετοχές και κάρτες δημιουργήσαμε την συνάρτηση **participation\_and\_cards**.

```

def participation_and_cards(list):

...
...

```

Αρχικά, διαχωρίζει τους τερματοφύλακες από τους υπόλοιπους παίκτες και τους ανακατεύει έτσι ώστε σε κάθε ματς να διαλέγονται τυχαία οι βασικοί και οι αναπληρωματικοί. Επίσης, για κάθε αγώνα διαχωρίζει αυτούς που θα παίζουν σε όλο το ματς με αυτούς που είτε θα είναι βασικοί και θα βγουν αλλαγή, είτε θα μπουν ως αλλαγή. Το **noch** δείχνει τον αριθμό των αλλαγών που θα κάνει ένας προπονητής και είναι ένας τυχαίος αριθμός από 1 έως 3.

```

...
for (id, pos) in list:

    if pos == "τερματοφύλακας":
```

```

        keeper.append(id)

    else:
        other.append(id)

random.shuffle(keeper)
random.shuffle(other)

play_all_match = other[0 : (basikoi - noch)]
play_all_match.append(keeper[0])
changes = other[(basikoi-noch) : ((basikoi-noch) + noch*2)]
...

```

- **Για τους παίκτες που παίζουν όλο τον αγώνα :**

Αγωνίζονται από minute\_in = 0 έως minute\_out = 90 εκτός εάν έχουν δεχτεί κόκκινη κάρτα, όπου το minute\_out ισούται με το λεπτό στο οποίο δέχτηκαν την κόκκινη.

- **Για τους παίκτες που είναι ή θα βγαντούν αλλαγή :**

Για κάθε έναν παίκτη που βγαίνει αλλαγή σε ένα λεπτό υπάρχει ένας άλλος που εισέρχεται στον αγωνιστικό χώρο στο ίδιο λεπτό.

```

for plr in play_all_match:

    if has_red_card and plr == play_all_match[0]:
        red_card_min = random.randint(5,85)
        participation.append(
            (plr, 0, red_card_min )
        )
        cards.append(
            (plr, 2, red_card_min)
        )
    else:
        participation.append(
            (plr, 0, 90)
        )

for i in range(0, noch):
    min = random.randint(40, 85)
    participation.append(
        ( changes[i] , 0, min )
    )
    participation.append(
        ( changes[i + noch], min, 90)
    )

for i in range(0, random.randint(0, max_yellow)):
    (id, min_in, min_o) = participation[random.randint(0, len(participation)-1)]
    if not id in have_yellow:
        have_yellow.append(id)

```

```

        cards.append(
            (id, 1, random.randint(min_in, min_o))
        )
    return {
        "keep": keeper,
        "other": other,
        "part_tbl": participation,
        "cards": cards
    }
}

```

Επίσης, στο τέλος η συνάρτηση επιστρέφει 4 πίνακες. Έναν που δείχνει τους τερματοφύλακες της ομάδας (**keep**), έναν που δείχνει τους υπόλοιπους παίκτες που έχουν δικαιώμα συμμετοχής στον αγώνα (**other**), έναν που δείχνει τους παίκτες που έχουν συμμετοχή καθώς και τα λεπτά συμμετοχής τους (**part\_tbl**) και έναν που δείχνει τους παίκτες που έχουν δεχτεί κάρτα (**cards**).

Για την δημιουργία δεδομένων για τα γκολ δημιουργήσαμε την συνάρτηση **goal\_achivers**:

Η συνάρτηση, αυτή, επιστρέφει έναν πίνακα **goals** που περιέχει τα id των ποδοσφαιριστών που σκοράρουν, τον τύπο του γκολ τους και το λεπτό που το πετυχαίνουν.

- **Για τους παίκτες που πετυχαίνουν τα γκολ στους αγώνες :**

Μας ενδιαφέρει να έχουν συμμετοχή στο τυχαίο λεπτό που θα πετύχουν το γκολ (να είναι δηλαδή ένα λεπτό ανάμεσα στο **minute\_in** και το **minute\_out** του παίκτη). Επίσης, τα γκολ της γηπεδούχου ομάδας (**goalsHome**) τα πετυχαίνουν είτε παίκτες της ίδιας της ομάδας και είναι γκολ υπέρ, είτε παίκτες της φιλοξενούμενης ομάδας και είναι αυτογκόλ. Αντίστοιχα και για τα γκολ της φιλοξενούμενης ομάδας (**goalsGuest**).

Η υλοποίηση της συνάρτησης **goal\_achivers**:

```

def goal_achivers(home_participation, guest_participation, goalsHome, goalsGuest):
    goals = []
    goalMinutes = []
    for i in range(0, goalsHome):
        is_own_goal = False
        if random.randint(0,100) < owngoal_pc:
            is_own_goal = True
        if is_own_goal:
            (id,      min_in,      min_out)      =      guest_participation[random.randint(0,
len(guest_participation)-1)]
            minute = rand_int_in_range(min_in, min_out, goalMinutes)
            goals.append(
                (id, 2, minute)
            )
            goalMinutes.append(minute)
        else:
            (id,      min_in,      min_out)      =      home_participation[random.randint(0,
len(home_participation)-1)]
            minute = rand_int_in_range(min_in, min_out, goalMinutes)
            goals.append(

```

```

        (id, 1, minute)
    )
    goalMinutes.append(minute)

for i in range(0, goalsGuest):
    is_own_goal = False
    if random.randint(0,100) < ownGoal_pc:
        is_own_goal = True

    if is_own_goal:
        (id, min_in, min_out) = home_participation[random.randint(0,
len(home_participation)-1)]
        minute = rand_int_in_range(min_in, min_out, goalMinutes)
        goals.append(
            (id, 2, minute)
        )
        goalMinutes.append(minute)
    else:
        (id, min_in, min_out) = guest_participation[random.randint(0,
len(guest_participation)-1)]
        minute = rand_int_in_range(min_in, min_out, goalMinutes)
        goals.append(
            (id, 1, minute)
        )
        goalMinutes.append(minute)

return goals

```

Αυτές, λοιπόν, τις δύο συναρτήσεις που είδαμε παραπάνω θα τις χρησιμοποιήσουμε για να δημιουργήσουμε τα στατιστικά δεδομένα για τους ποδοσφαιριστές.

Αυτό που πρέπει να κάνουμε αρχικά είναι να διαβάσουμε από την βάση μας όλους τους ποδοσφαιριστές όλων των ομάδων που δημιουργήσαμε. Για να το κάνουμε, αυτό, θα χρησιμοποιήσουμε πάλι την συνάρτηση **ReadSQL** που έχουμε προαναφέρει. **Ομος**, από τον 2<sup>ο</sup> γύρο και μετά δεν θα πρέπει να συλλέγουμε όλους τους παίκτες αλλά μόνο όσους έχουν δικαίωμα συμμετοχής στον εν λόγω αγώνα, δηλαδή, αυτούς που δεν έχουν πάρει κόκκινη κάρτα στην προηγούμενη αγωνιστική. Για αυτό πρέπει να συντάσσουμε δύο ερωτήματα SQL, ένα για την πρώτη αγωνιστική και ένα για όλες τις υπόλοιπες. Αυτά τα ερωτήματα υλοποιούνται στην παρακάτω συνάρτηση. Παρατηρούμε ότι στο δεύτερο query χρησιμοποιούμε ένα υποερώτημα για να αφαιρέσουμε τους τιμωρημένους ποδοσφαιριστές.

```

def playersOfTeam(id_tps, round): # id_team_per_season

    validPlayersSQL_Round1 = '''
        SELECT
            tp.id_player, p.position
        FROM TEAM_PLAYER tp
        LEFT JOIN PLAYER p ON p.id_player = tp.id_player
        WHERE tp.id_team_per_season = ?
    '''

```

```

validPlayersSQL = '''

SELECT
    tp.id_player, p.position
FROM TEAM_PLAYER tp
LEFT JOIN PLAYER p ON p.id_player = tp.id_player
WHERE tp.id_team_per_season = ?
AND tp.id_player NOT IN ( -- players with red card in the prev round
    SELECT tp.id_player
    FROM TEAM_PLAYER tp
    LEFT JOIN PLAYER_PARTICIPATION pp ON pp.id_player = tp.id_player
    LEFT JOIN MATCH m ON m.id_match = pp.id_match
    LEFT JOIN MATCHWEEK mw ON mw.id_matchweek = m.id_matchweek
    LEFT JOIN PLAYER_CARD pc ON pc.id_player = tp.id_player AND pc.id_match =
m.id_match
    WHERE (m.id_team_home = ? OR m.id_team_guest = ?)
    AND mw.round = ? -- (round - 1)
    AND pc.id_card = ? -- type = 2 = red-card
)
'''

if round == 1:
    return readSQL(database, validPlayersSQL_Round1, (id_tps, ))
else:
    return readSQL(database, validPlayersSQL, (id_tps, id_tps, id_tps, round-1, 2))

```

Συνεπώς, για κάθε ομάδα και για κάθε αγωνιστική μπορούμε να αντλούμε τους διαθέσιμους ποδοσφαιριστές. Επιπλέον, κάθε αγώνας έχει ένα σκορ το οποίο χρειάζεται να ξέρουμε για να δημιουργήσουμε τους σκόρερ για κάθε ομάδα. Αυτό υπολογίζεται τυχαία από τους πίνακες:

<code>goalH = [0,1,1,1,1,2,2,2,2,3,3,3,4,4,5]</code>
<code>goalG = [0,0,1,1,1,1,1,2,2,2,2,3,3,4]</code>

Παρατηρούμε ότι η εντός έδρας ομάδα έχει παραπάνω πιθανότητες για νίκη από ότι η εκτός έδρας κάτι που είναι και λογικό σε ένα πραγματικό πρωτάθλημα.

Για την εισαγωγή των δεδομένων στους πίνακες:

```

def run_matches(season, league, round):
    mw_id = (season * 10 + league) * 100 + round
    print('Matchweek:', mw_id)
    roundMatches = readSQL(database, '''
        SELECT id_team_home, id_team_guest
        FROM MATCH WHERE id_matchweek = ?
    ''', (mw_id, ))
    idxi = 0
    for match in roundMatches:

```

```

homePlayers = playersOfTeam(match[0], round)
guestPlayers = playersOfTeam(match[1], round)
goalsHome = randItem(goalH)
goalsGuest = randItem(goalG)
idxi = idxi + 1
id_match = mw_id * 10 + idxi
home_p_and_cards = participation_and_cards(homePlayers)
away_p_and_cards = participation_and_cards(guestPlayers)
goals_per_match = goal_achivers(home_p_and_cards["part_tbl"],
away_p_and_cards["part_tbl"], goalsHome, goalsGuest)

```

Παρατηρούμε ότι αυτή η συνάρτηση δέχεται ως ορίσματα το id της season, το id της κατηγορίας και το round και υπολογίζει τα στατιστικά στοιχεία για όλα τα ματς μιας συγκεκριμένης αγωνιστικής.

Το μόνο που μένει να γίνει είναι να γραφτούν τα δεδομένα στην βάση. Για να γραφτούν θα χρησιμοποιούμε την συνάρτηση **writeSQL()**.

- Σημείωση για το **id\_participation** :

Είναι ένας αριθμός 8 ψηφιών όπου τα 5 πρώτα ψηφία δείχνουν το **id\_match** και τα τελευταία 3 το **id\_player** που έχει συμμετοχή.

**Παράδειγμα:** id\_participation → 11021041 : Ο παίχτης με id 41 έχει συμμετοχή στον αγώνα με id 11021, δηλαδή τον 1<sup>o</sup> αγώνα της 2<sup>ης</sup> αγωνιστικής της πρώτης κατηγορίας.

Για τους πίνακες **PLAYER\_CARD, PARTICIPATION, PLAYER\_PARTICIPATION**:

```

for i in range(0, len(home_p_and_cards["part_tbl"])) :
    id_p = home_p_and_cards["part_tbl"][i][0]
    id_participation = id_match * 1000 + id_p
    writeSQL(database, '''

        INSERT INTO PARTICIPATION
(id_participation, minute_in, minute_out
)
VALUES  (?, ?, ?)
''', (
        id_participation,
        home_p_and_cards["part_tbl"][i][1],
        home_p_and_cards["part_tbl"][i][2]
))

    writeSQL(database, '''

        INSERT INTO PLAYER_PARTICIPATION
id_match, id_player, id_participation
)
VALUES  (?, ?, ?)
''', (
        id_match,
        id_p,
        id_participation
))

```

```

        ))
for i in range(0,len(home_p_and_cards["cards"])) :
    writeSQL(database, """
        INSERT INTO PLAYER_CARD (id_match,id_player,id_card,minute
        )
VALUES  (?, ?, ?, ?)
"""
, (
        id_match,
        home_p_and_cards["cards"][i][0],
        home_p_and_cards["cards"][i][1],
        home_p_and_cards["cards"][i][2]
    ))

```

Αντίστοιχα και για την εκτός έδρας ομάδα ( 2 ίδια loops με **away\_p\_and\_cards** πίνακα)

Για τον πίνακα **PLAYER\_GOAL**:

```

for i in range(0,len(goals_per_match)) :
    writeSQL(database, """
        INSERT INTO PLAYER_GOAL (id_match,id_player,id_goal,minute
        )
VALUES  (?, ?, ?, ?)
"""
, (
        id_match,
        goals_per_match[i][0],
        goals_per_match[i][1],
        goals_per_match[i][2]
    ))

```

Τέλος για να μην τρέξουμε αυτήν την συνάρτηση για 3(κατηγορίες) x 22 (rounds) = 66 φορές δημιουργούμε την συνάρτηση, την οποία εκτελούμε μόνο τρεις φορές, μια για κάθε κατηγορία.

```

def run_matchweeks(season, league):
    for i in range(1,23):
        run_matches(season, league, i)
        print('αγωνιστική :', i)

```

Και με αυτόν τον τρόπο, δημιουργούμε μια βάση με πάρα πολλά δεδομένα. Συγκεκριμένα, στον πίνακα **PARTICIPATION**, **PLAYER\_PARTICIPATION** έχουμε πάνω από **9000 δεδομένα** και στους πίνακες **PLAYER\_CARD** και **PLAYER\_GOAL** πάνω από **1500**.

**Όλα αυτά τα αρχεία της python υπάρχουν και σε αρχείο zip για να μπορούν να μελετηθούν αναλυτικά.**

#### 4 ΜΕΘΟΔΟΛΟΓΙΑ – ΕΝΤΟΛΕΣ CRUD (CREATE-READ-UPDATE-DELETE)

Εφόσον δημιουργήσαμε τη βάση μας, διατυπώσαμε κάποιες εντολές SQL που επιτρέπουν αλλαγή στο περιεχόμενό της. Αρχικά, υλοποιήσαμε τις συναρτήσεις CRUD (CREATE-READ-UPDATE-DELETE). Οι συναρτήσεις θα δέχονται ως είσοδο ένα λεξικό και θα επιστρέψουν το SQL ερώτημα που θα πρέπει να εφαρμόσουμε στην βάση μας, για να μπορούμε να κάνουμε τις απαραίτητες τροποποιήσεις.

Συνάρτηση για **insert(create)** :

```
def insert(query):
    table, field, value = itemgetter('table', 'field', 'value')(query)
    sql = f"INSERT INTO \'{table}\' "
    fList = ', '.join(field)
    sql += f"({fList})"
    valList = ', '.join(value)
    sql += f"\nVALUES ({valList})"
    return sql
```

Δέχεται ως είσοδο ένα λεξικό με 3 κλειδιά. Το table από το οποίο αντλούμε τον πίνακα στον οποίο θέλουμε να εισάγουμε τα δεδομένα, το field από το οποίο παίρνουμε τα γνωρίσματα στα οποία θέλουμε να δώσουμε νέες τιμές και το value το οποίο εμπεριέχει τις νέες πλειάδες που θέλουμε να εισάγουμε στην βάση μας. Ένα παράδειγμα, για κατανόηση της συνάρτησης insert :

```
q = {
    "table": 'USER',
    "field": [
        'username', 'password'
    ],
    "value": [
        "Andreas", "0000-1-2-3-4"
    ]
}
print(insert(q))
```

Το οποίο επιστρέφει :

```
Programs/Python/Python310/python.exe "c:/Users/andre/OneDrive/Εγγραφα/ΣΧΟΛΗ -HM_TOC/crud.py"
INSERT INTO "USER" (username, password)
VALUES ("Andreas", "0000-1-2-3-4")
```

Με την ίδια λογική δημιουργούμε και τις συναρτήσεις **read, update, delete**.

Συνάρτηση για **read** :

```
def read(query):
    table, field, where, order = itemgetter('table', 'field', 'where', 'order')(query)
    fList = "*"
    if field != None:
        fList = ', '.join(field)
    sql = f"SELECT {fList}\n"
    sql += f"FROM {table}\n"
    whereStatement = ""
    if where != None:
        whereItems = []
        for (fname, operator, val) in where:
```

```

        whereItems.append(f"{fname} {operator} {val}")

    whereStatement = f"WHERE {' AND '.join(whereItems)}"

    # order-by is not implemented

    sql += whereStatement

    return sql

```

Δέχεται ως είσοδο ένα λεξικό με 4 κλειδιά. Το table από το οποίο παίρνουμε τον πίνακα, το field από το οποίο επιλέγουμε τα γνωρίσματα, το where το οποίο εμπεριέχει τις συνθήκες με τις οποίες θα επιλέξουμε τις πλειάδες και το order για την σειρά εμφάνισης τους (δεν καλύπτεται στην συνάρτηση). Έτσι δημιουργούμε ένα ερώτημα της μορφής :

**SELECT ... FROM ... WHERE <συνθήκη1> AND <συνθήκη2>.**

Συνάρτηση για update :

```

def update( query ):

    table, sets, where = itemgetter('table', 'set', 'where')(query)

    sql = f"UPDATE \"{table}\"\\nSET "
    setStatements = []

    for (f, v) in sets:
        setStatements.append(f"{f} = {v}")

    sql += ', '.join(setStatements)
    whereStatement = ""

    if where != None:
        whereItems = []
        for (fname, operator, val) in where:
            whereItems.append(f"{fname} {operator} {val}")

        whereStatement = f"\\nWHERE {' AND '.join(whereItems)}"

    sql += whereStatement

    return sql

```

Δέχεται ως είσοδο ένα λεξικό με 3 κλειδιά. Το table από το οποίο αντλούμε τον πίνακα τον οποίο θέλουμε να τροποποιήσουμε, το set από το οποίο παίρνουμε τα γνωρίσματα τα οποία θέλουμε να αλλάξουν και τις νέες τιμές τους και το where το οποίο εμπεριέχει τις συνθήκες με τις οποίες θα επιλέξουμε ποιες πλειάδες θέλουμε διαφοροποιήσουμε. Έτσι δημιουργούμε ένα ερώτημα της μορφής :

**UPDATE ... SET <γνώρισμα> = <νέα τιμή> WHERE <συνθήκη1> AND <συνθήκη2>.**

Συνάρτηση για delete :

```

def delete(query):

    table, where = itemgetter('table', 'where')(query)

    sql = f"DELETE FROM \"{table}\""
    whereStatement = ""

    if where != None:
        whereItems = []
        for (fname, operator, val) in where:
            whereItems.append(f"{fname} {operator} {val}")

        whereStatement = f" WHERE {' AND '.join(whereItems)}"

    sql += whereStatement

    return sql

```

```

        whereStatement = f"\nWHERE { ' AND '.join(whereItems)}"

    sql += whereStatement

    return sql

```

Δέχεται ως είσοδο ένα λεξικό με 2 κλειδιά. Το table από το οποίο αντλούμε τον πίνακα τον οποίο θέλουμε να διαγράψουμε κάποιο στοιχείο και το where το οποίο εμπεριέχει τις συνθήκες με τις οποίες θα επιλέξουμε ποιες πλειάδες θα διαγράψουμε. Έτσι δημιουργούμε ένα ερώτημα της μορφής :

**DELETE FROM ... WHERE <συνθήκη1> AND <συνθήκη2>.**

Για να ενσωματώσουμε όλες αυτές τις λειτουργίες, θεωρούμε ότι στην εφαρμογή μας έχουμε έναν **Admin**. Αυτός με την σειρά του μπορεί να δημιουργεί (**insert**) κάποιους **roster\_admin** οι οποίοι είναι υπεύθυνοι για την εγγραφή νέων παικτών στην βάση και την εισαγωγή σε ομάδα. Επίσης, ο **Admin** θα έχει την δυνατότητα να διαγράφει κάποιον **roster\_admin** (**delete**). Επιπλέον, κάθε **Admin** και **roster\_admin** θα έχουν ένα μοναδικό username και ένα password, το οποίο θα μπορούν να το αλλάξουν ανά πάσα στιγμή (**update**), με το οποίο θα μπορούν να κάνουν login στην εφαρμογή. Τέλος, όπως και οι απλοί users, θα μπορούν να διαβάσουν (**read**) όλη την θεματολογία που παρέχει η εφαρμογή.

Έτσι καλύπτονται όλων των ειδών αλλαγές που μπορούμε να κάνουμε στην βάση μας.

Παρακάτω θα δούμε ένα παράδειγμα στο οποίο περιγράφονται όλα αυτά ( **Πιο αναλυτικά στην περιγραφή της γραφικής διεπαφής** ):

Αρχικά έχουμε δημιουργήσει στην βάση μας έναν πίνακα USER και έχουμε προσθέσει τον admin της εφαρμογής.

SELECT	*	FROM	USER	▼	Schema	Query Editor	Auto Re
			username varchar(30) NOT NULL PRIMARY KEY DEFAULT		password varchar(30) NOT NULL DEFAULT	role varchar(30) NOT NULL DEFAULT	
			Admin		1234	admin	

Μενού επιλογών απλού χρήστη (user) :

```

0. Exit
1. Check roster
...
7. Statistika
99. Login
Enter selection: 

```

Login ως Admin :

```

Enter selection: 99
Enter username: Admin
Enter password: 1234
choose what to do
0. Exit
1. Check roster
...
7. Statistika
10. insert player to roster
20. change password
21. create user
22. delete user
99. Logout

```

Παρατηρούμε και όλο το μενού επιλογών που έχει ένας admin.

Επιλογή 21 για δημιουργία roster\_admin :

```

choose what to do
0. Exit
1. Check roster
...
7. Statistika
10. insert player to roster
20. change password
21. create user
22. delete user
99. Logout
Enter selection: 21
Enter username of new roster_admin: alfa
Enter password of new roster_admin: 123

```

Με το που δημιουργηθεί ο roster\_admin ενημερώνεται η βάση :

SELECT * FROM USER			Schema	Query Editor	Auto Re
	username	password	role	varchar(30) NOT NULL	DEFAULT
1	Admin	1234	admin		
2	alfa	123	roster_admin		

Τώρα αν κάνουμε login στην εφαρμογή ως roster\_admin, θα δούμε ένα παρόμοιο μενού επιλογών στο οποίο το μόνο που έχει αφαιρεθεί είναι η δυνατότητα προσθήκης και διαγραφής κάποιου χρήστη.

```

Enter selection: 99
Enter username: alfa
Enter password: 123
choose what to do
0. Exit
1. Check roster
...
7. Statistika
10. insert player to roster
20. change password
99. Logout
Enter selection: 20
Enter new password (DO NOT FORGET): 123321

```

Επιλογή 20 για αλλαγή κωδικού. Ο νέος κωδικός ενημερώνεται στην βάση :

SELECT * FROM USER			Schema	Query Editor	Auto Re
	username	password	role	varchar(30) NOT NULL	DEFAULT
1	Admin	1234	admin		
2	alfa	123321	roster_admin		

Επιλογή 10 για την δημιουργία παίκτη και την εγγραφή σε ομάδα. Συμπληρώνουμε όλα τα γνωρίσματα του παίκτη, καθώς και την ομάδα στην οποία πρόκειται να αγωνιστεί. Αυτό που χρειάζεται να προσέξουμε είναι να μην του δώσουμε κάποιο αριθμό φανέλας που έχει κάποιος συμπαίκτης του.

```

Select Team id: 20

    create new player!
Player name: Andreas
last name: Mendrinos
birthday (yyyy-mm-dd): 2003-06-21
Nationality (* = Ellada): *
position: (* = mesos) *
non available shirts: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18
shirt number: 20

```

Και ο νέος παίκτης εισάγεται στην βάση στους πίνακες PLAYER και TEAM\_PLAYER:

653 762	Άννα	Καγιάφα	1998-12-12	Ελλάδα	99	μέσος
654 763	Andreas	Mandrinos	2003-06-21	Ελλάδα	20	μέσος

653	3	id_season:1 id...	762	name:'Avva' ...	2023-09-01	2024-05-31
654	20	id_season:1 i...	763	name:'Andrea...	2023-09-01	2024-05-31

Τέλος με Logout, βλέπουμε τις αρχικές επιλογές, δηλαδή αυτές που έχει ένας απλός χρήστης.

```

choose what to do
0. Exit
1. Check roster
...
7. Statistika
10. insert player to roster
20. change password
99. Logout
Enter selection: 99
choose what to do
0. Exit
1. Check roster
...
7. Statistika
99. Login

```

Με Exit κλείνουμε την εφαρμογή.

Όλα αυτά όπως προαναφέρθηκε, έχουν ενσωματωθεί στο γραφικό περιβάλλον.

## 5 ΜΕΘΟΔΟΛΟΓΙΑ – ΕΡΩΤΗΜΑΤΑ ΣΤΗ ΒΑΣΗ

Εφόσον δημιουργήσαμε τη βάση μας, διατυπώσαμε σε sqlite ερωτήματα προς αυτή για να λάβουμε διάφορα στατιστικά στοιχεία για τους παίκτες και τις ομάδες, πληροφορίες για τους αγώνες, τα αποτελέσματα και τη βαθμολογία. Τα ερωτήματα, αυτά, διατυπώθηκαν αρχικά στο **DB Browser for sqlite**, ώστε να ελέγχουμε τη σωστή λειτουργία τους και στη συνέχεια τα συμπεριλάβαμε στο πρόγραμμα σε python που γράφαμε για να δημιουργήσουμε την εφαρμογή μας. Πιο συγκεκριμένα, υλοποιήσαμε τα παρακάτω ερωτήματα:

- Για κάθε ομάδα θέλουμε όνομα ομάδας, γήπεδο, αριθμός ποδοσφαιριστών, κατηγορία, προπονητές:

```

SELECT t.id_team_per_season, cl.name AS team_name, a.name AS arena_name,
l.name, (SELECT COUNT(DISTINCT tp_inner.id_player)

FROM TEAM_PLAYER AS tp_inner

WHERE tp_inner.id_team_per_season = t.id_team_per_season) AS
player_count, GROUP_CONCAT(c.coach_info, ', ') AS coaches
FROM (((TEAM_PER_SEASON AS t JOIN CLUB AS cl ON t.EPO_id = cl.EPO_id)
JOIN LEAGUE AS l ON l.id_league = t.id_league) JOIN ARENA AS a ON
t.id_arena = a.id_arena) JOIN (SELECT DISTINCT id_team_per_season, name
|| ' ' || lastname || '(' || start_date || ' - ' || IFNULL(end_date,
'Present') || ')' AS coach_info FROM COACH) AS c ON t.id_team_per_season
= c.id_team_per_season

WHERE player_count <> 0
GROUP BY t.id_team_per_season
ORDER BY t.id_team_per_season;

```

- Ρόστερ ομάδων, δηλαδή για κάθε ομάδα θέλουμε ονόματα ποδοσφαιριστών, έτος γέννησης, θέση:

```

SELECT cl.name, p.id_player, p.name, p.lastname, strftime('%Y',
p.birthday) AS year_of_birth, p.position
FROM (((TEAM_PER_SEASON AS t JOIN CLUB AS cl ON cl.EPO_id = t.EPO_id)
JOIN TEAM_PLAYER AS tp ON t.id_team_per_season = tp.id_team_per_season)
JOIN PLAYER AS p ON tp.id_player = p.id_player)
ORDER BY cl.name, p.lastname;

```

- Θέλουμε το αποτέλεσμα των αγώνων, δηλαδή το id του αγώνα, τα id των ομάδων που αγωνίζονται, καθώς και τα γκολ υπέρ και γκολ κατά (αυτογκόλ) που βάζει κάθε ομάδα. Στο

σημείο, αυτό, αποφασίσαμε το συγκεκριμένο ερώτημα να μετατραπεί σε View, ώστε τα αποτελέσματά του να μπορούμε να τα αξιοποιήσουμε και σε επόμενα ερωτήματα και να μην χρειαστεί να τα υπολογίσουμε ξανά. Οπότε, ο κώδικας που αναπτύχθηκε είναι ο ακόλουθος:

```

CREATE VIEW MATCH_RESULT AS
SELECT m.id_match, m.id_team_home, m.id_team_guest, COALESCE((SELECT
count(pg.id_goal)                               FROM ((PLAYER_GOAL as pg JOIN GOAL as g on
pg.id_goal = g.id_goal) JOIN TEAM_PLAYER as tp on tp.id_player =
pg.id_player)

WHERE tp.id_team_per_season      =
m.id_team_home AND g.type = "γκολ_υπέρ" AND pg.id_match = m.id_match
GROUP BY pg.id_match),0)           as
goal_home_over, COALESCE((SELECT count(pg1.id_goal)
                           FROM ((PLAYER_GOAL as pg1 JOIN GOAL as g1 on
pg1.id_goal = g1.id_goal) JOIN TEAM_PLAYER as tp1 on tp1.id_player =
pg1.id_player)

WHERE tp1.id_team_per_season = m.id_team_home
AND g1.type = "αυτογκόλ" AND pg1.id_match = m.id_match
GROUP BY pg1.id_match),0)           as
goal_home_against, COALESCE((SELECT count(pg2.id_goal)
                           FROM ((PLAYER_GOAL as pg2 JOIN GOAL as g2 on
pg2.id_goal = g2.id_goal) JOIN TEAM_PLAYER as tp2 on tp2.id_player =
pg2.id_player)

WHERE tp2.id_team_per_season      =
m.id_team_guest AND g2.type = "γκολ_υπέρ" AND pg2.id_match = m.id_match
GROUP BY pg2.id_match),0)           as
goal_guest_over, COALESCE((SELECT count(pg3.id_goal)
                           FROM ((PLAYER_GOAL as pg3 JOIN GOAL as g3 on
pg3.id_goal = g3.id_goal) JOIN TEAM_PLAYER as tp3 on tp3.id_player =
pg3.id_player)

WHERE tp3.id_team_per_season      =
m.id_team_guest AND g3.type = "αυτογκόλ" AND pg3.id_match = m.id_match
GROUP BY pg3.id_match),0)           as
goal_guest_against
FROM MATCH as m

```

- Θέλουμε να δημιουργήσουμε το πρόγραμμα των αγώνων. Πιο συγκεκριμένα, για τους αγώνες σε όλες τις αγωνιστικές όλων των κατηγοριών θέλουμε τις εκάστοτε ομάδες που αγωνίζονται, το γήπεδο που διεξάγεται ο αγώνας (είναι έδρα μίας από τις δύο ομάδες), ημέρα και ώρα αγώνα, το σκορ και τους διαιτητές. Για να υπολογίσουμε το σκορ, μετράμε όλα τα γκολ υπέρ και αυτογκόλ κάθε μίας από τις δύο ομάδες, τα οποία αντλούμε από το View που δημιουργήσαμε στο προηγούμενο ερώτημα. Ο κώδικας που αναπτύχθηκε είναι ο παρακάτω:

```

SELECT l.name, ml.round, cl1.name as team_home, cl2.name as team_guest,
a.name as arena_name, m.match_time, m.match_date, ((mr.goal_home_over +
mr.goal_guest_against) || ' - ' || (mr.goal_guest_over +

```

```

mr.goal_home_against)) as score, m.referee, m.assistant_referee1,
m.assistant_referee2
FROM (((((MATCH as m JOIN MATCHWEEK as ml on m.id_matchweek =
ml.id_matchweek) JOIN league as l on ml.id_league = l.id_league) JOIN
TEAM_PER_SEASON as t on t.id_team_per_season = m.id_team_home) JOIN CLUB
as cl1 on cl1.EPO_id = t.EPO_id) join TEAM_PER_SEASON as t1 on
t1.id_team_per_season = m.id_team_guest) JOIN CLUB as cl2 on cl2.EPO_id =
t1.EPO_id) JOIN ARENA as a on a.id_arena = t.id_arena) JOIN MATCH_RESULT
as mr on mr.id_match = m.id_match);

```

- Θέλουμε να καταμετρούμε τις ποινές. Πιο συγκεκριμένα, για όσες ομάδες έχουν δεχτεί ποινή να φαίνεται το όνομα και το EPO\_id τους, η αιτία της ποινής, η διάρκειά της (είναι καθαρά θεωρητική στη βάση μας και ουσιαστικά σχετίζεται με το χρονικό διάστημα που η ομάδα θα αγωνίζεται κεκλεισμένων των θυρών, το οποίο όμως εμείς εδώ δεν εξετάζουμε), το χρηματικό πρόστιμο, οι βαθμοί ποινής (που αφαιρούνται από τη βαθμολογία της ομάδας) και η ημερομηνία που επιβλήθηκε η εκάστοτε ποινή. Ο κώδικας που αναπτύχθηκε είναι ο παρακάτω:

```

SELECT cl.name, cl.EPO_id, p.reason, p.duration, p.money_penalty,
p.points_penalty, tp.punishment_date
FROM(((TEAM_PER_SEASON as t JOIN CLUB as cl on t.EPO_id = cl.EPO_id) JOIN
TEAM_PUNISHMENT as tp on tp.id_team_per_season = t.id_team_per_season) JOIN
PUNISHMENT as p on p.id_punishment = tp.id_punishment) JOIN LEAGUE as l on l.id_league = t.id_league)
ORDER BY cl.name;

```

- Θέλουμε να παίρνουμε σε κάθε κατηγορία ποιες ομάδες ανήκουν. Το ερώτημα αυτό, ωστόσο, δεν το συμπεριλάβαμε στην εφαρμογή μας, διότι όλες τις πληροφορίες για τις ομάδες τις παίρνουμε έτσι κι αλλιώς ανά κατηγορία. Ωστόσο, ο κώδικας που αναπτύχθηκε είναι ο παρακάτω:

```

SELECT l.name, GROUP_CONCAT(' (' || t.id_team_per_season || ' , ' || t.EPO_id || ' , ' || cl.name || ')') as teams
FROM ((TEAM_PER_SEASON as t join CLUB as cl on t.EPO_id = cl.EPO_id) JOIN
LEAGUE as l on t.id_league = l.id_league)
GROUP BY l.name

```

- Θέλουμε τη βαθμολογία των ομάδων ανά κατηγορία και σεζόν. Πιο συγκεκριμένα, θέλουμε το id και το όνομα των ομάδων, τους αγώνες στους οποίους έχει λάβει μέρος κάθε ομάδα, τους πόντους της, τις νίκες, τις ήττες, τις ισοπαλίες και τα συνολικά γκολ υπέρ και αυτογκόλ που έχει σημειώσει στους αγώνες που έχει συμμετάσχει. Αξίζει εδώ να σημειωθεί ότι για κάθε νίκη έχουμε 3 πόντους, για κάθε ισοπαλία έχουμε 1 πόντο, για κάθε ήττα κανέναν και αφαιρούνται οι πόντοι από τις ποινές, όπως συμβαίνει και σε ένα πραγματικό τοπικό πρωτάθλημα ποδοσφαίρου. Στο σημείο, αυτό, αποφασίσαμε το συγκεκριμένο ερώτημα να μετατραπεί σε View, ώστε τα αποτελέσματά του να μπορούμε δυνητικά να τα αξιοποιήσουμε και σε επόμενα

ερωτήματα και να μην χρειαστεί να τα υπολογίσουμε ξανά. Ο κώδικας, λοιπόν, που αναπτύχθηκε είναι ο ακόλουθος:

```

CREATE VIEW RANKING AS
WITH MatchStats AS (
    SELECT
        t.id_team_per_season AS id,
        -- Νίκες
        SUM(CASE
            WHEN mr.id_team_home = t.id_team_per_season
                AND (mr.goal_home_over + mr.goal_guest_against) >
                    (mr.goal_guest_over + mr.goal_home_against) THEN 1
            WHEN mr.id_team_guest = t.id_team_per_season
                AND (mr.goal_home_over + mr.goal_guest_against) <
                    (mr.goal_guest_over + mr.goal_home_against) THEN 1
            ELSE 0
        END) AS win,
        -- Ήττες
        SUM(CASE
            WHEN mr.id_team_home = t.id_team_per_season
                AND (mr.goal_home_over + mr.goal_guest_against) <
                    (mr.goal_guest_over + mr.goal_home_against) THEN 1
            WHEN mr.id_team_guest = t.id_team_per_season
                AND (mr.goal_home_over + mr.goal_guest_against) >
                    (mr.goal_guest_over + mr.goal_home_against) THEN 1
            ELSE 0
        END) AS defeat,
        -- Ισοπαλίες
        SUM(CASE
            WHEN (mr.id_team_home = t.id_team_per_season OR
                mr.id_team_guest = t.id_team_per_season)
                AND (mr.goal_home_over + mr.goal_guest_against) =
                    (mr.goal_guest_over + mr.goal_home_against) THEN 1
            ELSE 0
        END) AS draw,
        -- Σύνολο Αγώνων
        COUNT(mr.id_match) AS matches,
        -- Γκολ Υπέρ
        SUM(CASE
            WHEN mr.id_team_home = t.id_team_per_season THEN
                (mr.goal_home_over + mr.goal_guest_against)
            WHEN mr.id_team_guest = t.id_team_per_season THEN
                (mr.goal_guest_over + mr.goal_home_against)
        END)
)
```

```

        ELSE 0
    END) AS goal_over,
-- Гкоλ Каtá
SUM(CASE
        WHEN mr.id_team_home = t.id_team_per_season THEN
(mr.goal_guest_over + mr.goal_home_against)
        WHEN mr.id_team_guest = t.id_team_per_season THEN
(mr.goal_home_over + mr.goal_guest_against)
        ELSE 0
    END) AS goal_against
FROM TEAM_PER_SEASON AS t
LEFT JOIN MATCH_RESULT AS mr ON t.id_team_per_season IN
(mr.id_team_home, mr.id_team_guest)
GROUP BY t.id_team_per_season
),
Penalties AS (
SELECT
    tp.id_team_per_season,
    SUM(p.points_penalty) AS points_penalty
FROM TEAM_PUNISHMENT AS tp
JOIN PUNISHMENT AS p ON tp.id_punishment = p.id_punishment
GROUP BY tp.id_team_per_season
),
TeamInfo AS (
SELECT
    t.id_team_per_season AS id,
    cl.name AS name,
    l.name AS league1,
    s.year
FROM TEAM_PER_SEASON AS t
JOIN CLUB AS cl ON t.EPO_id = cl.EPO_id
JOIN LEAGUE AS l ON t.id_league = l.id_league
JOIN SEASON AS s ON t.id_season = s.id_season
)
SELECT
    ti.year,
    ti.league1,
    ti.id,
    ti.name,
    ms.matches,
    (3 * ms.win + ms.draw - COALESCE(p.points_penalty, 0)) AS points,
    ms.win,
    ms.defeat,

```

```

        ms.draw,
        ms.goal_over,
        ms.goal_against
    FROM TeamInfo AS ti
    LEFT JOIN MatchStats AS ms ON ti.id = ms.id
    LEFT JOIN Penalties AS p ON ti.id = p.id_team_per_season
    ORDER BY ti.year, ti.league1, points DESC;

```

- Θέλουμε για κάθε σεζόν ανά κατηγορία τους 10 πρώτους σκόρερς, δηλαδή το id, το όνομα και το επίθετο κάθε παίκτη, τα συνολικά γκολ (όχι αυτογκόλ) που έχει σκοράρει και την ομάδα στην οποία αγωνίζεται. Οι παίκτες πρέπει να εμφανίζονται σε φθίνουσα σειρά τερμάτων. Ο κώδικας που αναπτύχθηκε είναι ο παρακάτω:

```

WITH RankedPlayers AS (
    SELECT
        s.year,
        l.name AS league_name,
        p.id_player,
        p.name AS player_name,
        p.lastname,
        COUNT(pg.id_goal) AS goals_over,
        cl.name AS club_name,
        ROW_NUMBER() OVER (PARTITION BY l.name ORDER BY COUNT(pg.id_goal)
DESC ,p.lastname) AS rank
    FROM PLAYER AS p
    JOIN PLAYER_GOAL AS pg ON p.id_player = pg.id_player
    JOIN GOAL AS g ON pg.id_goal = g.id_goal
    JOIN TEAM_PLAYER AS tp ON tp.id_player = p.id_player
    JOIN TEAM_PER_SEASON AS t ON t.id_team_per_season =
tp.id_team_per_season
    JOIN CLUB AS cl ON cl.EPO_id = t.EPO_id
    JOIN LEAGUE AS l ON l.id_league = t.id_league
    JOIN SEASON AS s ON s.id_season = t.id_season
    WHERE g.type = "γκολ_υπέρ"
    GROUP BY s.year, l.name, p.id_player, p.name, p.lastname, cl.name)

SELECT
    year,
    league_name,
    id_player,
    player_name,
    lastname,

```

```

    goals_over,
    club_name
FROM RankedPlayers
WHERE rank <= 10
ORDER BY league_name, rank;

```

- Θέλουμε για κάθε σεζόν ανά κατηγορία τους παίκτες με τις περισσότερες κίτρινες κάρτες, δηλαδή το id, το όνομα και το επίθετο κάθε παίκτη, τις συνολικές κίτρινες κάρτες που έχει δεχτεί και την ομάδα στην οποία αγωνίζεται. Οι παίκτες πρέπει να εμφανίζονται σε φθίνουσα σειρά κίτρινων καρτών. Ο κώδικας που αναπτύχθηκε είναι ο παρακάτω:

```

SELECT s.year, l.name, p.id_player, p.name, p.lastname, count(pc.id_card)
as yellow_cards, cl.name as club_name

FROM((((((PLAYER as p JOIN PLAYER_CARD as pc on p.id_player =
pc.id_player) JOIN CARD as c on pc.id_card = c.id_card) JOIN TEAM_PLAYER
as tp on tp.id_player = p.id_player)JOIN TEAM_PER_SEASON as t on
t.id_team_per_season = tp.id_team_per_season)JOIN CLUB as cl on cl.EPO_id
= t.EPO_id)JOIN LEAGUE as l on l.id_league = t.id_league)join season as
s on s.id_season = t.id_season)

WHERE c.type = "κίτρινη"

GROUP BY p.id_player

ORDER BY l.id_league, yellow_cards desc , p.lastname;

```

- Θέλουμε για κάθε σεζόν ανά κατηγορία τους παίκτες με τις περισσότερες κόκκινες κάρτες, δηλαδή το id, το όνομα και το επίθετο κάθε παίκτη, τις συνολικές κόκκινες κάρτες που έχει δεχτεί και την ομάδα στην οποία αγωνίζεται. Οι παίκτες πρέπει να εμφανίζονται σε φθίνουσα σειρά κόκκινων καρτών. Ο κώδικας που αναπτύχθηκε είναι ο παρακάτω:

```

SELECT s.year, l.name, p.id_player, p.name, p.lastname, count(pc.id_card)
as red_cards, cl.name as club_name

FROM((((((PLAYER as p JOIN PLAYER_CARD as pc on p.id_player =
pc.id_player) JOIN CARD as c on pc.id_card = c.id_card) JOIN TEAM_PLAYER
as tp on tp.id_player = p.id_player)JOIN TEAM_PER_SEASON as t on
t.id_team_per_season = tp.id_team_per_season)JOIN CLUB as cl on cl.EPO_id
= t.EPO_id)JOIN LEAGUE as l on l.id_league = t.id_league)join season as
s on s.id_season = t.id_season)

WHERE c.type = "κόκκινη"

GROUP BY p.id_player

ORDER BY l.id_league, red_cards desc, p.lastname;

```

- Θέλουμε για κάθε σεζόν ανά κατηγορία τους παίκτες με τα περισσότερα λεπτά συμμετοχής, δηλαδή το id, το όνομα και το επίθετο κάθε παίκτη, τα λεπτά συμμετοχής, το πλήθος των αγώνων στους οποίους έχει λάβει μέρος και την ομάδα στην οποία αγωνίζεται. Οι παίκτες πρέπει

να εμφανίζονται σε φθίνουσα σειρά λεπτών συμμετοχής. Ο κώδικας που αναπτύχθηκε είναι ο παρακάτω:

```

SELECT s.year, l.name, cl.name AS club_name, p.id_player, p.name,
p.lastname, sum(pa.minute_out - pa.minute_in) AS participation_minutes,
count(pp.id_match) AS matches

FROM((((((PLAYER AS p JOIN PLAYER_PARTICIPATION AS pp ON p.id_player =
pp.id_player) JOIN PARTICIPATION AS pa ON pp.id_participation =
pa.id_participation) JOIN TEAM_PLAYER AS tp ON tp.id_player =
p.id_player) JOIN TEAM_PER_SEASON AS t ON t.id_team_per_season =
tp.id_team_per_season) JOIN CLUB AS cl ON cl.EPO_id = t.EPO_id) JOIN LEAGUE
AS l ON l.id_league = t.id_league) JOIN season AS s ON s.id_season =
t.id_season)

GROUP BY p.id_player

ORDER BY l.id_league, participation_minutes DESC, p.lastname ASC;

```

- Θέλουμε κάθε σεζόν ανά κατηγορία για κάθε παίκτη κάθε ομάδας τους αγώνες στους οποίους έχει συμμετάσχει, δηλαδή το ονοματεπώνυμο του παίκτη, το id και την ημερομηνία του κάθε αγώνα, τα λεπτά συμμετοχής, τα γκολ υπέρ και τα αυτογκόλ που σκόραρε, τις κίτρινες και κόκκινες κάρτες που δέχτηκε στο συγκεκριμένο ματς, καθώς και το αν συμμετείχε στον αγώνα ως αλλαγή ή ως βασικός (αν το minute\_in είναι 0 είναι βασικός, αλλιώς μπαίνει ως αλλαγή). Στο σημείο, αυτό, αποφασίσαμε το συγκεκριμένο ερώτημα να μετατραπεί σε View, ώστε τα αποτελέσματά του να μπορούμε δυνητικά να τα αξιοποιήσουμε και σε επόμενα ερωτήματα και να μην χρειαστεί να τα υπολογίσουμε ξανά. Ο κώδικας, λοιπόν, που αναπτύχθηκε είναι ο ακόλουθος:

```

CREATE VIEW MATCHES_PLAYERS AS

SELECT
    s.year,
    l.name AS league_name,
    m.id_match,
    m.match_date,
    cl.name AS club_name,
    t.id_team_per_season,
    p.id_player,
    p.name AS player_name,
    p.lastname AS player_lastname,
    ((mr.goal_home_over + mr.goal_guest_against) || ' - ' || (mr.goal_guest_over + mr.goal_home_against)) AS score,
    (pa.minute_out - pa.minute_in) AS minutes_of_participation,
    SUM(CASE WHEN pg.id_goal = 1 THEN 1 ELSE 0 END) AS goals_over,
    SUM(CASE WHEN pg.id_goal = 2 THEN 1 ELSE 0 END) AS goals_against,
    SUM(CASE WHEN pc.id_card = 1 THEN 1 ELSE 0 END) AS yellow_cards,
    SUM(CASE WHEN pc.id_card = 2 THEN 1 ELSE 0 END) AS red_cards,

```

```

CASE
    WHEN pa.minute_in = 0 THEN 'βασικός'
    ELSE 'αλλαγή'
END AS player_status

FROM

MATCH AS m

JOIN MATCHWEEK AS mw ON m.id_matchweek = mw.id_matchweek
JOIN SEASON AS s ON mw.id_season = s.id_season
JOIN LEAGUE AS l ON l.id_league = mw.id_league
JOIN TEAM_PER_SEASON AS t ON (t.id_team_per_season = m.id_team_home
OR t.id_team_per_season = m.id_team_guest)
JOIN CLUB AS cl ON cl.EPO_id = t.EPO_id
JOIN TEAM_PLAYER AS tp ON tp.id_team_per_season =
t.id_team_per_season
JOIN PLAYER AS p ON p.id_player = tp.id_player
LEFT JOIN PLAYER_GOAL AS pg ON pg.id_player = p.id_player AND
pg.id_match = m.id_match
LEFT JOIN PLAYER_CARD AS pc ON pc.id_player = p.id_player AND
pc.id_match = m.id_match
JOIN MATCH_RESULT AS mr ON m.id_match = mr.id_match
JOIN PLAYER_PARTICIPATION AS pp ON pp.id_player = p.id_player AND
pp.id_match = m.id_match
JOIN PARTICIPATION AS pa ON pp.id_participation = pa.id_participation
GROUP BY
s.year,
l.name,
m.match_date,
m.id_match,
cl.name,
t.id_team_per_season,
p.id_player
ORDER BY
p.id_player--s.year, l.name, m.match_date, m.id_match, cl.name,
p.lastname;

```

- Θέλουμε για κάθε ποδοσφαιριστή στατιστικά για κάθε σεζόν. Πιο συγκεκριμένα, θέλουμε το id και το ονοματεπώνυμο του παίκτη, τα συνολικά γκολ και αυτογκόλ που έχει σκοράρει, τις συνολικές κάρτες, κόκκινες και κίτρινες, που έχει δεχτεί, τα συνολικά λεπτά συμμετοχής και τους συνολικούς αγώνες στους οποίους έχει συμμετάσχει σε μια σεζόν. Για να τα υπολογίσουμε όλα αυτά αξιοποιήσαμε το view που δημιουργήσαμε στο προηγούμενο ερώτημα. Οπότε, ο πίνακας που αναπτύχθηκε είναι ο παρακάτω:

```

SELECT mp.year, mp.id_player, mp.player_name, mp.player_lastname,
sum(mp.goals_over), sum(mp.goals_against), sum(mp.yellow_cards),
sum(mp.red_cards), sum(mp.minutes_of_participation), count(mp.id_match)
FROM MATCHES_PLAYERS as mp
GROUP BY mp.year, mp.id_player;

```

- Θέλουμε να καταγράφουμε τις ομάδες που πέφτουν κατηγορία, δηλαδή το όνομα των ομάδων, την κατηγορία από την οποία έπεσαν και την κατηγορία στην οποία βρέθηκαν. Οι ομάδες που πέφτουν κατηγορία είναι αυτές που βρίσκονται χαμηλότερα (η τελευταία ομάδα κάθε κατηγορίας) στην κατάταξη με βάση τη βαθμολογία τους. Αν περισσότερες από μία ομάδες έχουν την ίδια βαθμολογία, υποβιβάζεται η ομάδα με το μικρότερο goal difference (γκολ\_υπέρ – γκολ\_κατά), όπως συμβαίνει και σε μια πραγματική ΕΠΣ. Τέλος, υποβιβασμό μπορούν να υποστούν μόνο ομάδες που ανήκουν στην Α και στη Β κατηγορία. Οπότε, ο κώδικας που αναπτύχθηκε είναι ο ακόλουθος:

```

WITH RankedTeams AS (
    SELECT id, league1, name, ROW_NUMBER() OVER (PARTITION BY league1
    ORDER BY points ASC, (goal_over - goal_against) ASC, goal_over DESC) AS
    rank
    FROM RANKING
)
SELECT r.league1, GROUP_CONCAT(' (' || r.name || ' , ' || r.id || ')')
AS relegated_teams, (r.league1 || ' -> ' || COALESCE(l_next.name, '')) AS
leagues
FROM RankedTeams AS r
JOIN LEAGUE AS l1 ON r.league1 = l1.name
LEFT JOIN LEAGUE AS l_next ON l_next.id_league = l1.id_league + 1
WHERE r.rank = 1 AND r.league1 <> 'Γ'
GROUP BY r.league1, l1.id_league;

```

- Θέλουμε να καταγράφουμε τις ομάδες που ανεβαίνουν κατηγορία, δηλαδή το όνομα των ομάδων, την κατηγορία από την οποία ανέβηκαν και την κατηγορία στην οποία βρέθηκαν. Οι ομάδες που ανεβαίνουν κατηγορία είναι αυτές που βρίσκονται ψηλότερα (η πρώτη ομάδα κάθε κατηγορίας) στην κατάταξη με βάση τη βαθμολογία τους. Αν περισσότερες από μία ομάδες έχουν την ίδια βαθμολογία, ανεβαίνει η ομάδα με το μεγαλύτερο goal difference (γκολ\_υπέρ – γκολ\_κατά), όπως συμβαίνει και σε μια πραγματική ΕΠΣ. Τέλος, άνοδο μπορούν να πετύχουν μόνο ομάδες που ανήκουν στην Β και στη Γ κατηγορία. Οπότε, ο κώδικας που αναπτύχθηκε είναι ο ακόλουθος:

```

WITH RankedTeams AS (
    SELECT id, league1, name, ROW_NUMBER() OVER (PARTITION BY league1
    ORDER BY points DESC, (goal_over - goal_against) DESC, goal_over DESC)
    AS rank
    FROM RANKING
)

```

```

)
SELECT r.league1, GROUP_CONCAT(' (' || r.name || ' , ' || r.id || ')')
AS promoted_teams, (r.league1 || ' -> ' || COALESCE(l_next.name, ''))AS
leagues
FROM RankedTeams AS r
JOIN LEAGUE AS l1 ON r.league1 = l1.name
LEFT JOIN LEAGUE AS l_next ON l_next.id_league = l1.id_league - 1
WHERE r.rank = 1 AND r.league1 <> 'A'
GROUP BY r.league1, l1.id_league;

```

- Επειδή θέλαμε να αξιοποιήσουμε και την πράξη της διαίρεσης, αποφασίσαμε να βρούμε ανά κατηγορία τις ομάδες που έχουν σκοράρει τουλάχιστον 1 γκολ σε όλα τα εντός έδρας ματς. Ο κώδικας που αναπτύχθηκε είναι ο ακόλουθος:

```

SELECT DISTINCT l.name AS league_name, cl.name AS club_name
FROM (TEAM_PER_SEASON AS t JOIN CLUB AS cl ON t.EPO_id = cl.EPO_id JOIN
LEAGUE AS l ON t.id_league = l.id_league)
WHERE NOT EXISTS (
    SELECT 1
    FROM MATCH AS m
    WHERE (m.id_team_home = t.id_team_per_season) --OR m.id_team_guest =
t.id_team_per_season)
    AND NOT EXISTS (
        SELECT 1
        FROM MATCH_RESULT AS mr
        WHERE mr.id_match = m.id_match
        AND (
            (mr.id_team_home      =      t.id_team_per_season      AND
mr.goal_home_over > 0)
            /*OR
            (mr.id_team_guest      =      t.id_team_per_season      AND
mr.goal_guest_over > 0)*/
        )
    )
    AND EXISTS
    SELECT 1
    FROM MATCH AS m2
    WHERE m2.id_team_home = t.id_team_per_season OR m2.id_team_guest =
t.id_team_per_season
);

```

- Θέλουμε ανά κατηγορία να βρούμε τους παίκτες που έχουν βάλει γκολ από αλλαγή. Δηλαδή θέλουμε το id, το όνομα και το επίθετο του παίκτη που σκόραρε από αλλαγή, καθώς και το id του ματς στο οποίο αυτό συνέβη. Ο κώδικας που αναπτύχθηκε είναι ο παρακάτω:

```

SELECT      mp.year,      mp.id_match,      mp.league_name,      mp.id_player,
mp.player_name,  mp.player_lastname
FROM MATCHES_PLAYERS AS mp
WHERE mp.player_status = 'αλλαγή' AND mp.goals_over >= 1 --AND mp.score
<> '0 - 0'
GROUP BY mp.id_match;

```

- Θέλουμε ανά κατηγορία για τις ομάδες που έχουν κερδίσει σε πάνω από τους μισούς αγώνες στους οποίους έχουν συμμετάσχει, το ποσοστό από αυτές τις νίκες που επετεύχθησαν σε εντός έδρας αγώνες. Είναι ένα χρήσιμο ερώτημα για να παρατηρήσουμε αν οι ομάδες που έχουν αρκετές νίκες είναι πιο καλές στην έδρα τους. Ο κώδικας που αναπτύχθηκε είναι ο παρακάτω:

```

SELECT      r.league1,      r.id,      r.name, ((r.win * 100)/r.matches)    as
percentage_win,((select count(mr.id_match)
FROM MATCH_RESULT as mr
WHERE (mr.id_team_home = r.id and (mr.goal_home_over +
mr.goal_guest_against > mr.goal_guest_over + mr.goal_home_against))*
100/r.win) as percentage_home_win
FROM RANKING as r
GROUP BY r.id
HAVING ((r.win * 100)/r.matches)>=50;

```

## 6 ΜΕΘΟΔΟΛΟΓΙΑ – ΔΗΜΙΟΥΡΓΙΑ INDEXES

Μετά τη σύνταξη των queries αποφασίσαμε να δημιουργήσουμε μερικά χρήσιμα indexes, ώστε να επιταχυνθεί η αναζήτηση των δεδομένων που προκύπτει ως ανάγκη από την υποβολή των προαναφερθέντων ερωτημάτων. Η δημιουργία των indexes, λοιπόν, έγινε με κριτήριο τις συνενώσεις πινάκων και τις συνθήκες where που θέταμε σε καθένα από τα ερωτήματα, το feedback που λαμβάναμε από το “explain query plan” για κάθε select, αλλά και με γνώμονα ποια ερωτήματα μας ενδιαφέρει να επιταχύνουμε σε μεγαλύτερο βαθμό. Ταυτόχρονα, φροντίσαμε να υλοποιήσουμε και indexes που περιλαμβάνουν γνωρίσματα που λαμβάνουν μέρος στη δημιουργία των Views, ώστε να επιταχυνθούν σχετικά με αυτά τα Views ερωτήματα.

Παρατηρήσαμε, λοιπόν, ότι η δημιουργία των indexes επιτάχυνε σε μεγάλο βαθμό ερωτήματα, σε κάποια δε ο χρόνος εκτέλεσης μειώθηκε στο 1/3. Κάποια παραδείγματα χρόνων πριν και μετά την υλοποίηση των Indexes φαίνονται παρακάτω.

- Για το ερώτημα με το View “MATCHES\_PLAYERS”:

---

Execution finished without errors. Execution finished without errors.  
Result: 9905 rows returned in 330ms Result: 9905 rows returned in 176ms

Χρόνος εκτέλεσης πριν τη δημιουργία του index

Χρόνος εκτέλεσης μετά τη δημιουργία του index

- Για το ερώτημα με τα στατιστικά των παικτών:

Execution finished without errors. Execution finished without errors.  
Result: 648 rows returned in 284ms Result: 648 rows returned in 146ms

Χρόνος εκτέλεσης πριν τη δημιουργία του index

Χρόνος εκτέλεσης μετά τη δημιουργία του index

- Για το ερώτημα με τις κίτρινες κάρτες:

Execution finished without errors. Execution finished without errors.  
Result: 611 rows returned in 55ms Result: 611 rows returned in 9ms

Χρόνος εκτέλεσης πριν τη δημιουργία του index

Χρόνος εκτέλεσης μετά τη δημιουργία του index

## 7 ΔΗΜΙΟΥΡΓΙΑ ΕΦΑΡΜΟΓΗΣ – ΓΡΑΦΙΚΟ ΠΕΡΙΒΑΛΛΟΝ

Μετά και από τη δημιουργία των indexes, η βάση μας είχε στηθεί, οπότε έπρεπε να δημιουργήσουμε ένα γραφικό περιβάλλον, ώστε να μπορούν οι χρήστες να έχουν πρόσβαση στα δεδομένα της βάσης μας και να ενημερώνονται για τα θέματα του τοπικού πρωταθλήματος μέσω των αποτελεσμάτων των ερωτημάτων που αποφασίσαμε να θέσουμε σε αυτή.

Σε πρώτη φάση έπρεπε να αποφασίσουμε με ποιον ρόλο θα μπορεί κάθε χρήστης να αξιοποιήσει την εφαρμογή μας και ποιες δυνατότητες θα έχει ανάλογα με τον ρόλο που διαδραματίζει. Σκεφτήκαμε, λοιπόν, οι ρόλοι να είναι τρεις, αυτός του απλού “user”, αυτός του “roster\_admin”, κι αυτός του “admin”. Η ιεραρχία σχετικά με τις ενέργειες στις οποίες δύναται να προβεί κάθε χρήστης στην εφαρμογή μας είναι σύμφωνη με τη σειρά που αναφέρθηκαν οι ρόλοι (από τα λιγότερα στα περισσότερα προνόμια). Εδώ αξίζει να σημειωθεί ότι τις δυνατότητες που έχουν οι χρήστες που ανήκουν στην προηγούμενη βαθμίδα τις έχουν και εκείνοι που ανήκουν στην επόμενη συν κάποιες ακόμα. Ας αναφερθούμε, όμως, σε κάθε ρόλο με τη σειρά παραθέτοντας εκτός από τις δυνατότητές του και το γραφικό περιβάλλον με το οποίο έρχεται σε επαφή.

### 7.1 Ο ρόλος του user

Τον ρόλο του απλού user διαθέτει οποιοσδήποτε χρήστης ανοίγει την εφαρμογή μας και έχει πρόσβαση σε συγκεκριμένες λειτουργίες. Οι λειτουργίες αυτές αφορούν στοιχεία για τις ομάδες και τους παίκτες.

Πιο συγκεκριμένα, ο user μπορεί να ενημερωθεί για θέματα που αφορούν τις **ομάδες**:

- Χαρακτηριστικά ομάδων διαχωρισμένες στις κατηγορίες που ανήκουν

Επιλέξτε ερώτημα:

**Χαροκτηριστικά ομάδι**

**Έκτελεση**

Όνομα Ομάδας	Γήπεδο	Αριθμός Ποδοσφαιριστών	Προπονητές
Πέιρος Ιωάννης	Πέιρος Ιωάννης	18	Παναγιώτης Ιωάννου (2023-09-01 - 2024-05-31)
ΑΟ Δρεπάνου	ΑΟ Δρεπάνου	18	Στέφανος Δημητρίου (2023-09-01 - 2024-05-31)
ΑΟ Πάτρας	ΑΟ Πάτρας	18	Αντώνης Μηχαλίδης (2023-09-01 - 2024-05-31)
ΑΕ Ροΐτικου	ΑΕ Ροΐτικου	18	Βασίλης Αναγνωστόπουλος (2023-09-01 - 2024-05-31)
ΑΟ Σαραβαλίου	ΑΟ Σαραβαλίου	18	Ηλίας Παπακωνταντούνου (2023-09-01 - 2024-05-31)
ΑΟ Ζαβλανίου	ΑΟ Ζαβλανίου	18	Μάριος Γεωργίου (2023-09-01 - 2024-05-31)
ΑΟ Καστριταίου	ΑΟ Καστριταίου	18	Σωτήρης Κυριακίδης (2023-09-01 - 2024-05-31)
ΑΟ Μιντούλιον	ΑΟ Μιντούλιον	18	Περικλής Αλεξάνδρου (2023-09-01 - 2024-05-31)
ΑΟ Ριου	ΑΟ Ριου	18	Αλέξης Παπαδόπουλος (2023-09-01 - 2024-05-31)
ΑΟ Αγιάς	ΑΟ Αγιάς	18	Διονύσης Σταματόπουλος (2023-09-01 - 2024-05-31)
ΑΟ Βραχνέων	ΑΟ Βραχνέων	18	Μιχάλης Αναστασίου (2023-09-01 - 2024-05-31)
ΑΟ Κάτω Αχαΐας	ΑΟ Κάτω Αχαΐας	18	Γιάννης Μακρής (2023-09-01 - 2024-05-31)

Επιλέξτε κατηγορία:

**A**

- Πρόγραμμα αγώνων των ομάδων διαχωρισμένες στις κατηγορίες που ανήκουν

Επιλέξτε ερώτημα:

**Πρόγραμμα αγώνων**

**Έκτελεση**

Αγωνιστική	Γηπεδούχος	Φιλοξενούμενος	Γήπεδο	Ώρα αγώνα	Ημερομηνία	Σκορ	Διαιτητής	Α'βοηθός	Β' βοηθός
1	Πέιρος Ιωάνης	ΑΟ Κάτω Α'	Πέιρος Ιωάνης	16:00	2023-09-08	3 - 1	Ιωάννης Νικ.	Ζήνωνας Χρ.	Κώστας Τρι
1	ΑΟ Δρεπάνου	ΑΟ Βραχνέων	ΑΟ Δρεπάνου	16:00	2023-09-08	4 - 3	Γιώργος Πατ Νότης Μιχ.	Τόνο Μάρκ.	
1	ΑΟ Πάτρας	ΑΟ Αγιάς	ΑΟ Πάτρας	16:00	2023-09-08	2 - 1	Ιάσονωνας Στ. Μανούσας	Δημήτρης Γ.	
1	ΑΕ Ροΐτικου	ΑΟ Ριου	ΑΕ Ροΐτικου	16:00	2023-09-08	2 - 2	Δημήτρης Ο	Μίκης Πατ.	Βασίλειος Μ.
1	ΑΟ Σαραβαλίου	ΑΟ Μιντούλιον	ΑΟ Σαραβαλίου	16:00	2023-09-08	2 - 3	Δημήτρης Μ	Πίπτης Πατ.	Μάριος Κων.
1	ΑΟ Ζαβλανίου	ΑΟ Καστριταίου	ΑΟ Ζαβλανίου	16:00	2023-09-08	2 - 0	Τόνο Παπαδ.	Πίπτης Ιωάν.	Στέλιος Νικ.
2	Πέιρος Ιωάνης	ΑΟ Βραχνέων	Πέιρος Ιωάνης	16:00	2023-09-15	4 - 0	Γώργος Ευ.	Τζήνης Χρισ.	Βασίλειος Στ.
2	ΑΟ Κάτω Α'	ΑΟ Αγιάς	ΑΟ Κάτω Α'	16:00	2023-09-15	3 - 2	Άγγελος Πατ.	Φιλίππος Τε.	Πέτρος Πατ.
2	ΑΟ Δρεπάνου	ΑΟ Ριου	ΑΟ Δρεπάνου	16:00	2023-09-15	1 - 0	Οφρέας Κων.	Πίπτης Μαρ.	Παναγιώτης
2	ΑΟ Πάτρας	ΑΟ Μιντούλιον	ΑΟ Πάτρας	16:00	2023-09-15	4 - 2	Πέτρος Ραφι.	Μαρίνος Αθ.	Κωνσταντίν
2	ΑΕ Ροΐτικου	ΑΟ Καστριταίου	ΑΕ Ροΐτικου	16:00	2023-09-15	3 - 3	Δημήτρης Αν.	Εμμανουήλ.	Κώστας Χατ.
2	ΑΟ Σαραβαλίου	ΑΟ Ζαβλανίου	ΑΟ Σαραβαλίου	16:00	2023-09-15	0 - 0	Οδυσσασ Αν.	Μανούσας Ι.	Μίλτος Κων.
3	Πέιρος Ιωάνης	ΑΟ Αγιάς	Πέιρος Ιωάνης	16:00	2023-09-22	3 - 1	Γώργος Πατ.	Γώργος Πα.	Παυσανίας Ι.
3	ΑΟ Βραχνέων	ΑΟ Ριου	ΑΟ Βραχνέων	16:00	2023-09-22	3 - 2	Αριστείδης Κ.	Ζήνωνας Χρ.	Γώργος Ια.
3	ΑΟ Κάτω Α'	ΑΟ Μιντούλιον	ΑΟ Κάτω Α'	16:00	2023-09-22	3 - 1	Παναγιώτης	Ηλίας Χατζ.	Ανδρέας Πα.
3	ΑΟ Δρεπάνου	ΑΟ Καστριταίου	ΑΟ Δρεπάνου	16:00	2023-09-22	1 - 2	Ανδρέας Αθ.	Γιάννης Γαβ.	Ομηρος Πα.
3	ΑΟ Πάτρας	ΑΟ Ζαβλανίου	ΑΟ Πάτρας	16:00	2023-09-22	1 - 0	Κωστας Πατ.	Νότης Αράτ.	Ηλίας Αντω.
3	ΑΕ Ροΐτικου	ΑΟ Σαραβαλίου	ΑΕ Ροΐτικου	16:00	2023-09-22	0 - 0	Λουκάς Αθαν.	Αρης Κωνσ.	Νίκος Παπα.

Επιλέξτε κατηγορία:

**B**

- Βαθμολογία των ομάδων στην κατηγορία τους

Ομάδες

Επιλέξτε ερώτημα:

Βαθμολογία

Εκτέλεση

Σεζόν	Ομάδα	Αγώνες	Βαθμοί	Νίκες	Ηττες	Ισοπαλίες	Γκολ Υπέρ	Γκολ Κατά
2023	Αρης Μυτιλί	22	38	11	6	5	43	29
2023	ΑΟ Εγνακόδι	22	36	10	6	6	40	29
2023	ΑΟ Παραλίας	22	33	10	9	3	45	45
2023	Μάχη Καστρι	22	33	10	9	3	46	48
2023	Αστέρας Πατ	22	30	8	8	6	35	40
2023	ΑΟ Ιωνικότας	22	29	7	7	8	35	42
2023	Θρησκας Σάρα	22	29	7	7	8	38	39
2023	ΑΟ Ριάλου	22	27	7	9	6	38	43
2023	Ελπίδα Ροττίκ	22	27	8	8	6	42	39
2023	ΑΟ Αροης	22	26	6	8	8	32	30
2023	Κεραυνος Ζάφ	22	24	5	8	9	40	43
2023	Δρέξι Δρεπαν	22	23	5	9	8	43	50

Επιλέξτε κατηγορία:

Επιλεγμένη

- **Ρόστερ των ομάδων.** Εδώ υπάρχει δυνατότητα επιλογής της ομάδας. Επιπλέον, αν πατήσει ο χρήστης πάνω σε κάποιον παίκτη έχει τη δυνατότητα να δει τους συνολικούς αγώνες στους οποίους έχει αγωνιστεί μέσα στη σεζόν, καθώς και τα συνολικά στατιστικά του

Ομάδες

Επιλέξτε ερώτημα:

Ρόστερ Ομάδων

Εκτέλεση

Id	Όνομα	Επίθετο	Έτος γέννησης	Θέση
50	Μαρίνος	Αλεξίου	1992	επιθετικός
41	Ζάχος	Αράπογλου	1993	ομυντικός
52	Χρόνης	Αράπογλου	1994	μέσος
764	Βελής	Γλυκούς	2003	επιθετικός
43	Πριάμος	Ιωάνου	2002	ομυντικός
46	Μανούσας	Κωνσταντίνου	2000	μέσος
47	Μίκης	Μακρής	2003	μέσος
51	Ιουλίος	Παπαμιχαήλ	1994	ομυντικός
53	Δημήτρης	Παπαπέτρου	1998	επιθετικός
55	Παναγιώτης	Παπαχρήστου	1996	επιθετικός
40	Ξενοφών	Σίγμα	1993	τερματοφύλακας
48	Θέμης	Σιμεών	1998	μέσος
44	Νεκτάριος	Σταματόπουλος	1992	ομυντικός
49	Ηλίας	Σταματόπουλος	2004	επιθετικός
57	Άκης	Σωτηρίου	2005	ομυντικός
56	Θάνος	Χατζηγιαβόης	1992	επιθετικός
45	Ζάχος	Χατζηιακώβου	1995	μέσος
42	Κωνσταντίνος	Χατζηκωνάνου	1995	ομυντικός

Επιλέξτε ομάδα:

Παναγοϊκή

Επιλέξτε ερώτημα:

Ρόστερ Ομάδων ▾

Εκτέλεση

Id	Όνομα	Επίθετο	Έτος γέννησης	Θάση
50	Μαρίνος	Αλέξου	1992	επιθετικός
41	Ζάχος	Αράπογλου	1993	εμυντικός
52	Χρόνης	Αράπογλου	1994	μέσος
764	Βέλης	Γλυκος	2003	επιθετικός
43	Πριάμος	Ιωάνου	2002	εμυντικός
46	Μανούσος	Κωνσταντίνου	2000	μέσος
47	Μικης	Μακρής	2003	μέσος
51	Ιούλιος	Παπαμαρτίνη	1994	εμυντικός
53	Δημήτρης	Παπαπέτρου	1998	επιθετικός
55	Παναγιώτης	Παπαχρήστου	1996	επιθετικός
40	Ξενοφών	Σήμα	1993	τερματοφύλακας
48	Θέμης	Σήμεν	1998	μέσος
44	Νεκτάριος	Σταυροπότουλος	1992	εμυντικός
49	Ηλίας	Σταυροπότουλος	2004	επιθετικός
57	Άκης	Σωτηρίου	2005	εμυντικός
56	Θεόνος	Χατζηγιαβάνης	1992	επιθετικός
45	Ζάχος	Χατζηλακώσου	1995	μέσος
42	Κωνσταντίνος	Χατζημανόνου	1995	εμυντικός

Επιλέξτε ενέργεια:

Αγώνες ▾

Στατιστικά

### Επιλογή ποδοσφαιριστή

Επιλέξτε ερώτημα:

Ρόστερ Ομάδων ▾

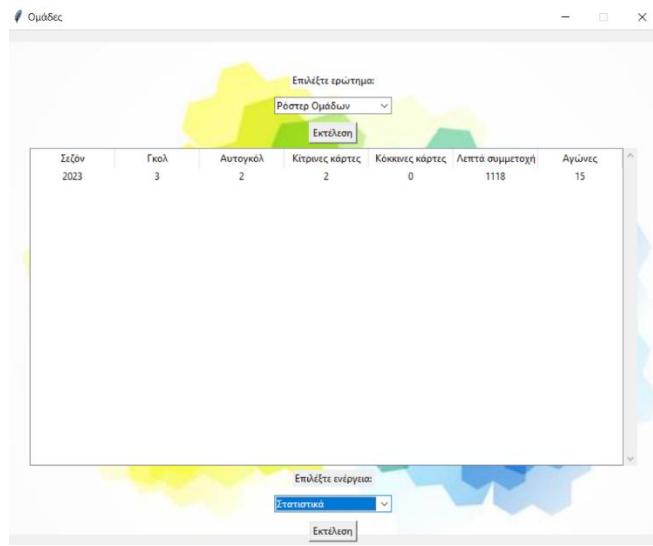
Εκτέλεση

Id αγώνα	Ημερομηνία	Σκορ	Λεπτά συμμετ.	Γκολ	Αυτογκόλ	Κίτρινες κάρτ	Κόκκινες κάρτ	Ρόλος
11011	2023-09-08	2 - 1	90	0	0	0	0	βασικός
11021	2023-09-15	1 - 0	90	0	0	0	0	βασικός
11041	2023-09-29	2 - 2	39	1	0	1	0	αλλαγή
11051	2023-10-06	3 - 1	90	1	0	0	0	βασικός
11061	2023-10-13	2 - 1	90	0	0	0	0	βασικός
11081	2023-10-27	1 - 2	90	0	0	0	0	βασικός
11091	2023-11-03	3 - 2	90	0	0	0	0	βασικός
11101	2023-11-10	2 - 2	90	1	1	0	0	βασικός
11121	2023-11-24	0 - 1	20	0	0	0	0	αλλαγή
11131	2023-12-01	4 - 1	58	0	0	1	0	βασικός
11161	2023-12-22	0 - 1	90	0	0	0	0	βασικός
11191	2024-01-12	1 - 2	90	0	0	0	0	βασικός
11201	2024-01-19	1 - 1	11	0	0	0	0	αλλαγή
11211	2024-01-26	3 - 0	90	0	0	0	0	βασικός
11221	2024-02-02	3 - 1	90	0	1	0	0	βασικός

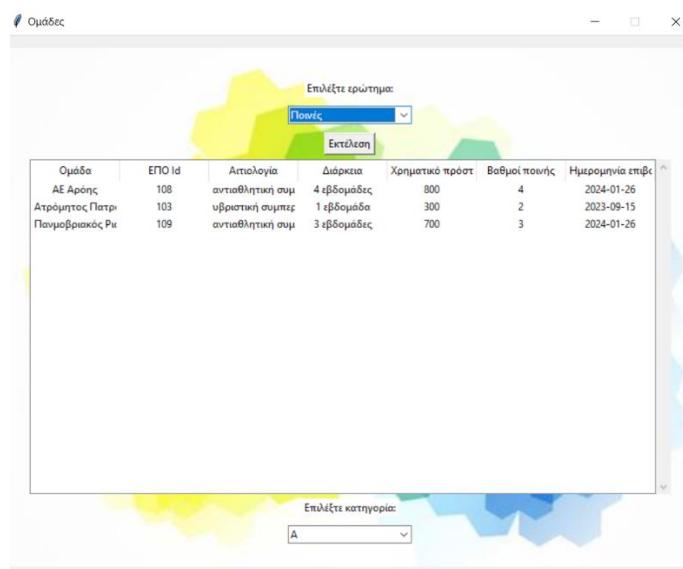
Επιλέξτε ενέργεια:

Αγώνες ▾

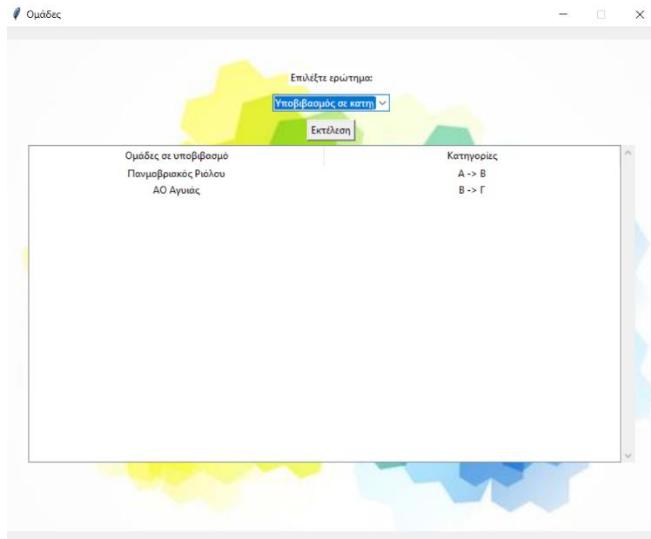
Εκτέλεση



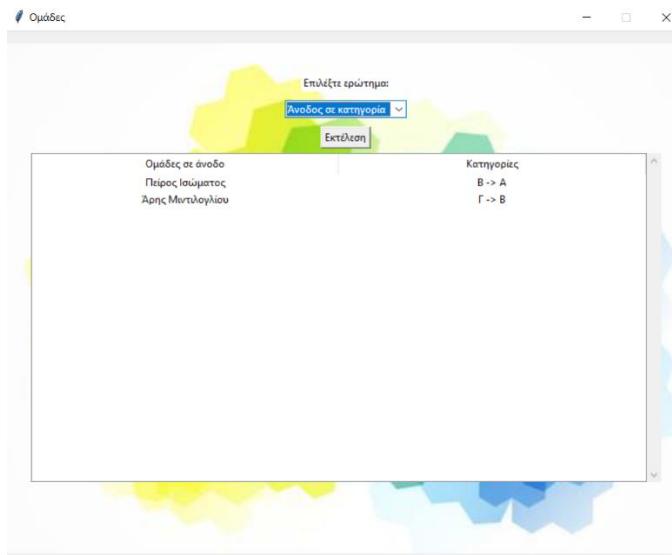
- Ποινές που έχουν λάβει ομάδες διάφορων κατηγοριών



- Ομάδες που έπεσαν κατηγορία



- Ομάδες που ανέβηκαν κατηγορία



Επιπλέον, ο user μπορεί να ενημερωθεί για top 10 παικτών σε διάφορες κατηγορίες:

- Top 10 scorers

Top 10 παικτών

Επιλέξτε ερώτημα:

Top 10 σκόρερς

Εκτέλεση

Σεζόν	Όνομα	Επίθετο	Γκολ	Ομάδα
2023	Ανδρέας	Παπακωνσταντίνου	7	Θύελλα Πατρών
2023	Μαρίνος	Χατζηρωτίου	6	Απόλλων Εγκυδας
2023	Φώτης	Σαράφογλου	6	Αστέρας Τσουκαλέκων
2023	Θάνος	Χατζηγιαβάνης	5	Παναχαϊκή
2023	Νίκος	Χατζηβασιλείου	5	Ατρόμιτος Πατρών
2023	Δημήτρης	Καραμιανώφ	5	Θύελλα Πατρών
2023	Λαζαρος	Μακρής	5	ΑΕ Αρόπης
2023	Νεκτάριος	Καλύμνου	5	ΑΕ Αρόπης
2023	Γιάννης	Παπαδόπουλος	5	Πανμοβιακός Ριώλου
2023	Δημήτρης	Καραγιάνης	5	Άχαική

Επιλέξτε κατηγορία:

A

- Top 10 παικτών σε κίτρινες κάρτες

Top 10 παικτών

Επιλέξτε ερώτημα:

Top 10 κίτρινων κάρτες

Εκτέλεση

Σεζόν	Όνομα	Επίθετο	Κίτρινες κάρτες	Ομάδα
2023	Μικής	Σωτηρίου	8	ΑΟ Άγιας
2023	Μάτιος	Στεργίου	7	ΑΟ Δρεπάνου
2023	Στέφανος	Πεγλιβανογλου	7	ΑΕ Ροΐτκων
2023	Γιώργος	Αθανασίου	7	ΑΟ Ζαβλανίου
2023	Ηλίας	Παπαβασιλείου	7	ΑΟ Μινιτογλίου
2023	Ιάσσωνας	Οικονόμου	6	ΑΟ Πάτρας
2023	Κώστας	Αναγνωστόπουλος	6	ΑΟ Πάτρας
2023	Όμηρος	Παπαχήρητου	6	ΑΟ Σαραβαλίου
2023	Στέλιος	Παπαβασιλείου	6	ΑΟ Σαραβαλίου
2023	Μάρκελος	Χατζηαποστόλου	6	ΑΟ Σαραβαλίου

Επιλέξτε κατηγορία:

B

- Top 10 παικτών σε κόκκινες κάρτες

Top 10 παικτών

Σεζόν	Όνομα	Επίθετο	Κόκκινες κάρτες	Ομάδα
2023	Ορφέας	Κυπριανού	2	Θησέας Σαραβαλίου
2023	Γιάννης	Ιωάνου	2	Κεραυνός Ζαβλανίου
2023	Βασίλειος	Χατζήβασιλείου	1	ΑΟ Ριόλου
2023	Πλάστινας	Μπουζιζάκης	1	ΑΟ Αρόπις
2023	Δημήτρης	Ιωάννου	1	ΑΟ Αρόπις
2023	Προκόπης	Παπακωνσταντίνου	1	ΑΟ Αρόπις
2023	Μίκης	Χατζηγεωργίου	1	ΑΟ Ιαώματος
2023	Τζώρτης	Χατζηματθαίου	1	ΑΟ Ιαώματος
2023	Σάκης	Τριανταφύλλου	1	ΑΟ Ιαώματος
2023	Λέων	Καραγεωργίου	1	ΑΟ Ιαώματος

Επιλέξτε ερώτημα:

Top 10 κόκκινων καρτών

Επιλέξτε κατηγορία:

Επιλέξτε ερώτημα:

Top 10 λεπτά συμμετοχής

Επιλέξτε κατηγορία:

- Top 10 παικτών σε λεπτά συμμετοχής

Top 10 παικτών

Σεζόν	Ομάδα	Όνομα	Επίθετο	Λεπτά συμμετοχής	Αγώνες που αγωνίστηκαν
2023	Αστέρας Τσουκαλέικη	Δημήτρης	Καρογιάνης	1741	21
2023	Ατρόμιτος Πατρών	Γιάννης	Δημητρίου	1653	19
2023	Θύελλα Πατρών	Παυσανίας	Ιωάννου	1648	20
2023	ΑΕ Αρόπις	Σώτος	Παπακωνσταντίνου	1602	21
2023	Αχαϊκή	Φωλιππός	Χατζησυατού	1588	19
2023	Απόλλων Εγνατίας	Στύρος	Αναστασίου	1585	19
2023	ΑΕ Αρόπις	Γιάννης	Διαμαντόπουλος	1576	19
2023	Αναγέννηση Πατρών	Μίκης	Μιχαηλίδης	1566	19
2023	Παναμερικαϊκός Ριόλο	Παυσανίας	Παπαδημητρίου	1554	19
2023	Παναμερικαϊκός Ριόλο	Τόννυ	Παπακωνσταντίνου	1521	20

Επιλέξτε ερώτημα:

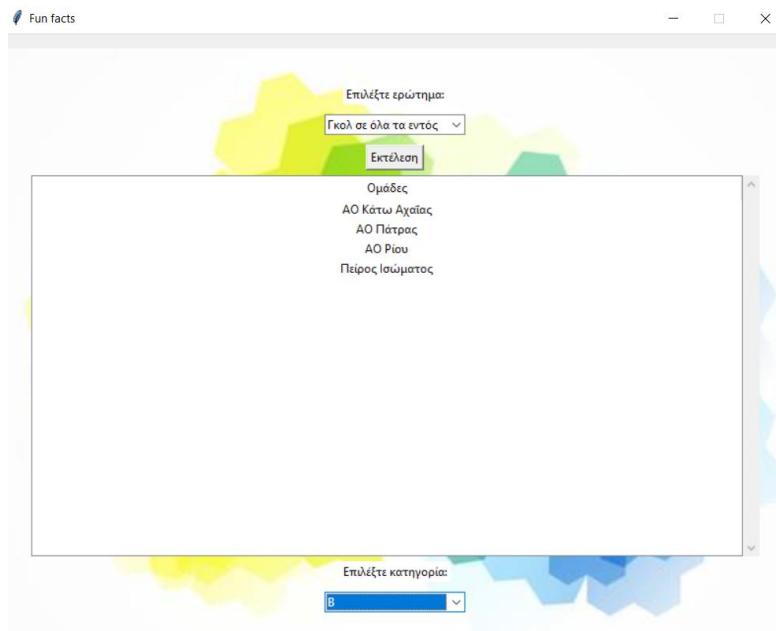
Top 10 λεπτά συμμετοχής

Επιλέξτε κατηγορία:

A

Τέλος, ο user μπορεί να ενημερωθεί για κάποια fun facts που αφορούν παίκτες ή ομάδες και αφορούν:

- Ομάδες που έχουν σκοράρει τουλάχιστον ένα γκολ σε όλα τα εντός έδρα παιχνίδια τους

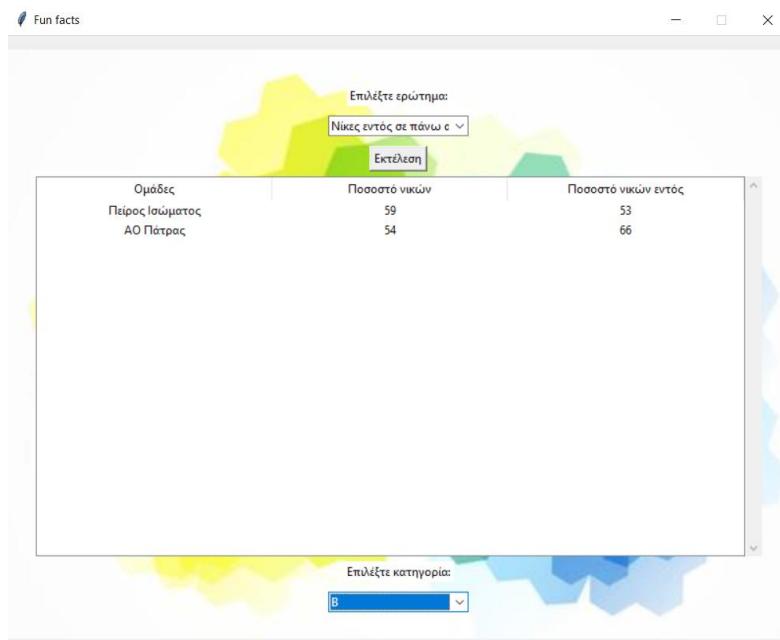


- Παίκτες που έχουν σκοράρει γκολ από αλλαγή

The screenshot shows a table of players who scored goals from substitutions. The columns are: Σεζόν, Ιδ ματς, Ιδ παικτή, Όνομα, and Επιβετο. The data is as follows:

Σεζόν	Ιδ ματς	Ιδ παικτή	Όνομα	Επιβετο
2023	12015	430	Κώστας	Αποστόλου
2023	12031	297	Δημήτρης	Παπαδόπουλος
2023	12032	449	Κώστας	Στάνκογλου
2023	12041	456	Λευτέρης	Γερβιδογιάνου
2023	12043	403	Φύλιππος	Χατζηματθαίου
2023	12044	396	Ομηρος	Μακρης
2023	12053	388	Κώστας	Σίγμα
2023	12062	392	Θεμιστοκλής	Αθανασίου
2023	12063	443	Πέτρος	Ραφαήλ
2023	12071	295	Ζάχος	Αθανασίου
2023	12074	450	Πανασσονίας	Χατζηγεωργίου
2023	12083	407	Ανέστης	Χατζηνικολάου
2023	12084	429	Δημήτρης	Σταματόπουλος
2023	12085	450	Πανασσονίας	Χατζηγεωργίου
2023	12086	486	Δημησθένης	Κυπρανού
2023	12094	407	Ανέστης	Χατζηνικολάου
2023	12096	475	Μίκης	Σωτηρίου
2023	12105	409	Μάνος	Φωτιού

- Πόσες (σε ποσοστό) είναι οι νίκες σε εντός έδρας ματς για ομάδες που έχουν νικήσει σε πάνω από τα μισά παιχνίδια στα οποία έχουν συμμετάσχει



## 7.2 Ο ρόλος του roster\_admin

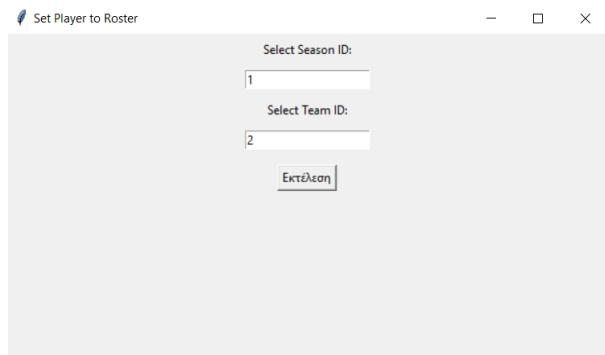
Ο ρόλος του roster\_admin απευθύνεται για παράδειγμα σε διαχειριστές των σωματείων που αγωνίζονται στο τοπικό. Με τον ρόλο αυτό κάποιος έχει πρόσβαση σε όσα έχει κι ένα απλός user, αλλά επιπλέον μπορεί να κάνει Log in με το Username και το Password του στην εφαρμογή μας και να προσθέσει κάποιον παίκτη σε ρόστερ ομάδας κάποιας σεζόν, αλλά και να αλλάξει το password του.

**Σημείωση:** Έχει δημιουργηθεί roster\_admin με username:alpha και password:1234

Οι προαναφερθείσες λειτουργίες φαίνονται παρακάτω:

Παράθυρο Login

Παράθυρο επιλογών (εδώ Εισαγωγή παίκτη)



Επιλογή ομάδας όπου θα ενταχθεί ο παίκτης

Create New Player

Non-available shirts: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18

Player Name	<input type="text" value="Γιώργος"/>
Last Name	<input type="text" value="Παπαδόπουλος"/>
Birthday (yyyy-mm-dd)	<input type="text" value="2000-10-26"/>
Nationality (*=Ελλάδα)	<input type="text" value="*"/>
Position (*=μέσος)	<input type="text" value="επιθετικός"/>
Shirt Number	<input type="text" value="19"/>

Εισαγωγή των στοιχείων του παίκτη

Success



Player created successfully!

OK

Success

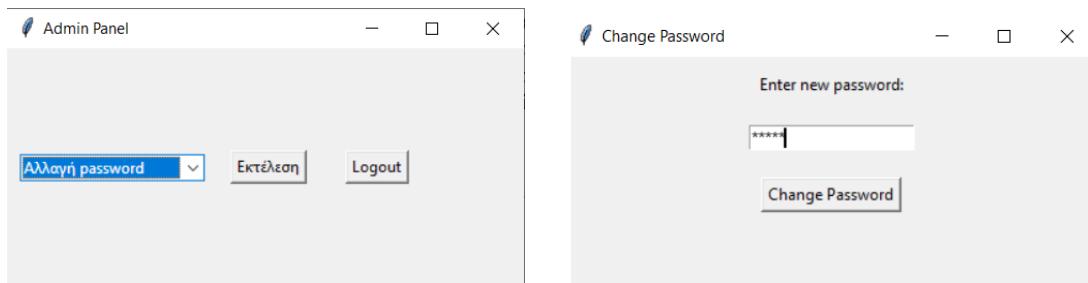


Player added to roster successfully!

OK

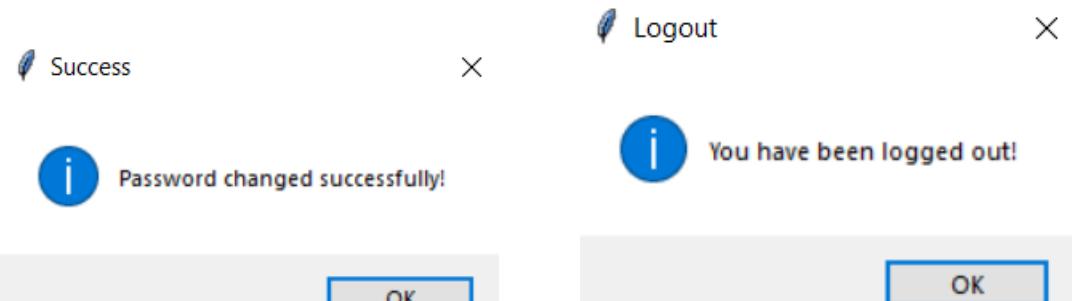
Μήνυμα επιτυχούς αποθήκευσης του παίκτη

Μήνυμα επιτυχούς εισαγωγής του παίκτη σε ομάδα



Επιλογή αλλαγής password

Αλλαγή password



Μήνυμα επιτυχούς καταχώρησης νέου password

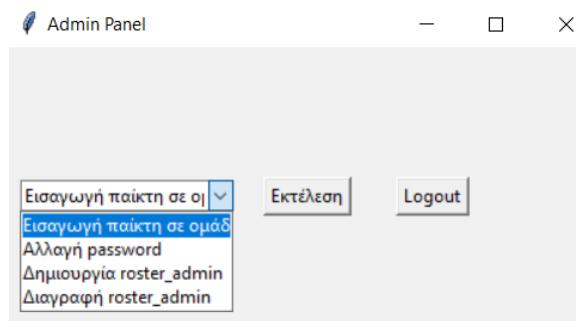
Μήνυμα επιτυχούς αποσύνδεσης

### 7.3 Ο ρόλος του admin

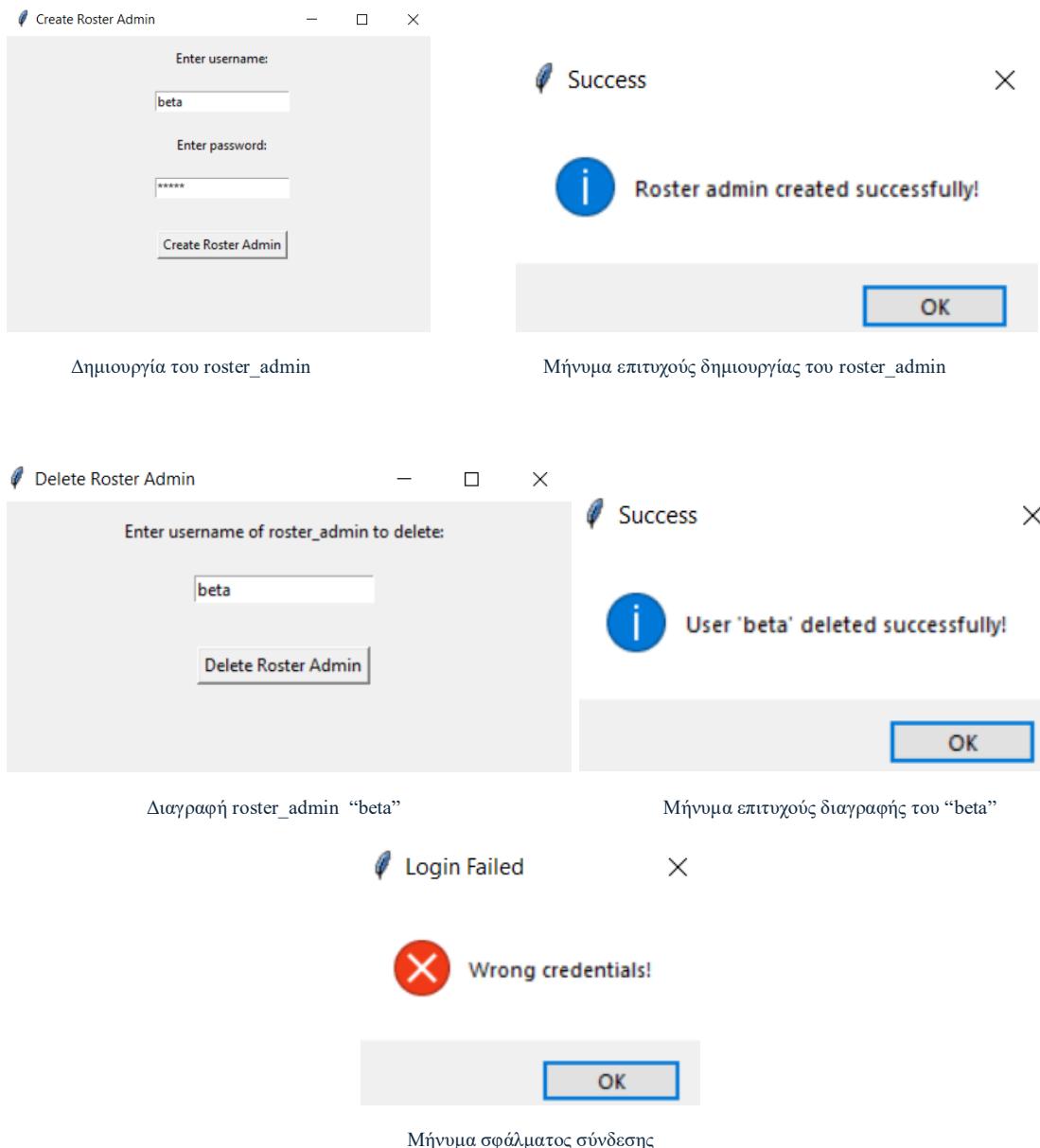
Ο ρόλος του admin απευθύνεται στον διαχειριστή της εφαρμογής. Ο admin, λοιπόν, σε αυτό το στάδιο που βρίσκεται η εφαρμογή μας (σύγουρα επιδέχεται βελτιώσεις) έχει τις ίδιες δυνατότητες με έναν roster\_admin, ενώ επιπλέον μπορεί να δημιουργήσει ή να διαγράψει έναν roster\_admin.

**Σημείωση:** Έχει δημιουργηθεί admin με username:Admin και password:1234

Οι επιπρόσθετες, αυτές, λειτουργίες φαίνονται παρακάτω:



Επιλογές του admin



## 8 ΔΙΑΜΟΙΡΑΣΜΟΣ ΕΡΓΑΣΙΩΝ

Το παρόν πρότζεκτ αποτελείται από πολλά σκέλη, με αποτέλεσμα να έχει αρκετό φόρτο εργασίας και να απαιτεί στενή και καλή συνεργασία για τη δημιουργία και τη σωστή διασύνδεσή τους. Συνεπώς, δουλέψαμε κάποια κομμάτια συνεργατικά δια ζώσης και άλλα κομμάτια εξ αποστάσεως αξιοποιώντας εργαλεία όπως το AnyDesk και το gitHub, ώστε να μπορούμε να μοιράσουμε δίκαια τις εργασίες και να συντονιστούμε σωστά για την ομαλή ολοκλήρωση της εργασίας.

Αν και απαιτείται, λοιπόν, στενή συνεργασία και ενασχόληση και των δύο σε όλα τα μέρη της εργασίας, ο καθένας επικεντρώθηκε σε περισσότερο σε συγκεκριμένα σκέλη. Ο διαμοιρασμός που ακολουθήσαμε είναι ο εξής:

- Δημιουργία ERD: από κοινού (διά ζώσης)
- Δημιουργία Schema: από κοινού (διά ζώσης)
- Δημιουργία βάσης και διασφάλιση αναφορικής ακεραιότητας: από κοινού (διά ζώσης)
- Εισαγωγή δεδομένων στη βάση με ai: A. Καγιάφα
- Εισαγωγή δεδομένων στη βάση με python: A. Μενδρινός
- Επιλογή SQL queries: από κοινού
- Διατύπωση SQL queries: A. Καγιάφα

- CRUD queries: Α. Μενδρινός
- Αξιολόγηση λειτουργιών CRUD: από κοινού
- Δημιουργία Indexes: Α. Μενδρινός
- Δημιουργία GUI εφαρμογής: Α. Καγιάφα

## 9 ΧΡΟΝΟΔΙΑΓΡΑΜΜΑ

Το χρονοδιάγραμμα που ακολουθήθηκε, ώστε να ολοκληρώσουμε την παρούσα εργασία είναι το παρακάτω:

- Δημιουργία ERD: Τέλη Οκτωβρίου
- Δημιουργία Schema: Τέλη Οκτωβρίου
- Δημιουργία βάσης και διασφάλιση αναφορικής ακεραιότητας: Αρχές Δεκεμβρίου
- Εισαγωγή δεδομένων στη βάση με ai: Αρχές Δεκεμβρίου
- Εισαγωγή δεδομένων στη βάση με python: Αρχές Δεκεμβρίου
- Επιλογή SQL queries: Αρχές Δεκεμβρίου
- Διατύπωση SQL queries: Διακοπές Χριστουγέννων
- CRUD queries: Διακοπές Χριστουγέννων
- Αξιολόγηση λειτουργιών CRUD: Διακοπές Χριστουγέννων
- Δημιουργία Indexes: Διακοπές Χριστουγέννων
- Δημιουργία GUI εφαρμογής: Διακοπές Χριστουγέννων

## 10 ΑΞΙΟΛΟΓΗΣΗ

Η αξιολόγηση της εργασίας μας στηρίχτηκε σε ορισμένα κριτήρια επιτυχίας, με τα οποία εξασφαλίζεται η αξιοπιστία και λειτουργικότητα της εφαρμογής μας. Πιο συγκεκριμένα:

- Αρχικά εργαστήκαμε σε βάση με λίγα δεδομένα, που κάλυπταν όμως τις ανάγκες των queries μας, ώστε να διασφαλίσουμε τη σωστή διατύπωσή τους και την ορθότητα των αποτελεσμάτων που προκύπτουν.
- Δόθηκε ιδιαίτερη βαρύτητα στη λογική σειρά εισαγωγής των δεδομένων στη βάση μας, ώστε να μην υπάρχουν ασάφειες και σφάλματα ιδιαίτερα σε δεδομένα που σχετίζονται μεταξύ τους. Για παράδειγμα, πρώτα εισήχθησαν τα λεπτά στα οποία ο παίκτης βρίσκεται στον αγωνιστικό χώρο και μετά οι κάρτες που δέχεται ή τα γκολ που σκοράρει, για να διασφαλιστεί ότι δίνονται σε χρόνο που εκείνος αγωνίζεται.
- Η εφαρμογή μας διαθέτει απλά γραφικά, που καθιστούν την πρόσβαση στα δεδομένα και την ανάλυσή τους από τον χρήστη εύκολη και προσιτή.
- Η εισαγωγή παικτών σε ομάδες από τον χρήστη πραγματοποιείται επιτυχώς, γεγονός που φαίνεται από τα μηνύματα που τυπώνονται και ο παίκτης φαίνεται κατευθείαν στο ρόστερ στο γραφικό περιβάλλον της εφαρμογής.

## 11 ΟΔΗΓΙΕΣ

### 11.1 Οδηγίες εγκατάστασης

Για να εκτελεστεί ορθά η εφαρμογή πρέπει να ακολουθηθούν τα παρακάτω βήματα:

**Προαπαιτούμενο:**

Εάν δεν έχετε εγκατεστημένη την Python στον υπολογιστή σας, μπορείτε να την εγκαταστήσετε από τον σύνδεσμο <https://www.python.org/downloads/>

Επίσης, θα χρειαστεί να ανοίξετε τερματικό και να εκτελέσετε την παρακάτω εντολή για να εγκαταστήσετε τη βιβλιοθήκη Pillow:

```
python -m pip install Pillow
```

Σε ορισμένα συστήματα Linux, μπορεί να χρειαστεί να εγκαταστήσετε τη βιβλιοθήκη tkinter με την ακόλουθη εντολή: sudo apt-get install python3-tk

**Βήματα εγκατάστασης:**

Ακολουθήστε τον σύνδεσμο: [https://github.com/annakag13/data\\_bases\\_project](https://github.com/annakag13/data_bases_project)

Κάντε κλικ στο πράσινο κουμπί Code και μετά Download ZIP.

Αποσυμπιέστε το αρχείο, μεταβείτε στον φάκελο championship\_app (στο zip αρχείο που υποβλήθηκε στο e-class ο σωστός φάκελος είναι championship\_app\_and\_base) και ανοίξτε ένα τερματικό στον συγκεκριμένο φάκελο.

Αφού ανοίξετε το τερματικό εκτελέστε τις παρακάτω εντολές:

```
python championship_app.py
```

### 11.2 Παραδείγματα χρήσης

Παραδείγματα χρήσης έχουν παρατεθεί στη παρουσίαση του γραφικού περιβάλλοντος παραπάνω. Ουσιαστικά ο χρήστης ανοίγει την εφαρμογή και βλέπει το αρχικό μενού που δίνει τις επιλογές που έχουν προαναφερθεί παραπάνω.

Επίσης, υπάρχει βίντεο στο zip αρχείο που κατεβάζει ο χρήστης, στο οποίο γίνεται επίδειξη της χρήσης της εφαρμογής.

### 11.3 Σύνδεσμοι για τον κώδικα που αναπτύχθηκε

Όλα τα αρχεία κώδικα που αναπτύχθηκαν υπάρχουν στον σύνδεσμο:  
[https://github.com/annakag13/data\\_bases\\_project](https://github.com/annakag13/data_bases_project)