

Εφαρμογή τοπικού πρωταθλήματος ποδοσφαίρου

ΟΜΑΔΑ 20

Ιωάννης Κατοίκος

Αριθμός Μητρώου: 1066493 Έτος: 4^o

Κωνσταντίνα Θανοπούλου

Αριθμός Μητρώου: 1066581 Έτος: 4^o

Περίληψη:

Το θέμα του πρότζεκτ μας αφορούσε τη δημιουργία μιας εφαρμογής ενός τοπικού πρωταθλήματος ποδοσφαίρου. Σε αυτή την εφαρμογή θα πρέπει να φαίνονται οι αγώνες του πρωταθλήματος, οι ομάδες και οι παίκτες της κάθε ομάδας, τα διάφορα στατιστικά τους ανά αγώνα, η βαθμολογία του πρωταθλήματος καθώς και άλλα επιπλέον ενδιαφέροντα στατιστικά, όπως ο πρώτος σκόρερ του πρωταθλήματος και άλλα. Για τη δημιουργία αυτής της εφαρμογής, ήταν αναγκαία η δημιουργία μίας βάσης δεδομένων, στην οποία θα μπορούμε να αποθηκεύουμε τα δεδομένα του τοπικού πρωταθλήματος καθώς και να τα αξιοποιούμε με κατάλληλο τρόπο. Για τη δημιουργία της βάσης, έπρεπε να δημιουργηθεί το ανάλογο διάγραμμα οντοτήτων-συσχετίσεων (ERD) και το σχεσιακό σχήμα (Relational Schema) μέσα από τα οποία να ικανοποιούνται οι κανόνες του μικρόκοσμου του τοπικού πρωταθλήματος που θέσαμε εμείς. Το ενδιαφέρον με αυτά ήταν ότι το θέμα του τοπικού πρωταθλήματος ποδοσφαίρου έχει αυστηρούς κανόνες και περιορισμούς τους οποίους καλούμασταν να ικανοποιήσουμε και παράλληλα να βγάζουμε με ικανοποιητικό τρόπο τα διάφορα πεδία της εφαρμογής μας. Ένα άλλο ιδιαίτερα σημαντικό στοιχείο του πρότζεκτ είναι η συλλογή των δεδομένων, όπου και βασιστήκαμε στη συλλογή πραγματικών δεδομένων από το πρωτάθλημα της Premier League 2020-21. Τέλος, η εφαρμογή που φτιάχτηκε δείχνει γραφικά τα διάφορα δεδομένα, ενώ ακόμη δίνεται και η δυνατότητα τροποποίησης τους. Τα παραπάνω υλοποιούνται με χρήση της Python και SQLite.

Μεθοδολογία:

Πριν αρχίσουμε την υλοποίηση του πρότζεκτ μας, αναζητήσαμε πληροφορίες για τη δομή ενός πρωταθλήματος ποδοσφαίρου, τους κανόνες ενός ποδοσφαιρικού αγώνα και πληροφορίες για το τι ενδιαφέρει τον κόσμο πάνω σε αυτό, δηλαδή τι θα ήθελε κάποιος να μάθει σχετικά με αυτό το πρωτάθλημα. Επιπλέον, μελετήσαμε κάποιους ποδοσφαιρικούς αγώνες προκειμένου να δούμε και στην πράξη πως εφαρμόζονται οι κανόνες αυτοί. Στην συνέχεια, έχοντας δει αυτά, προσπαθήσαμε να ορίσουμε τους δικούς μας κανόνες και περιορισμούς για τον μικρόκοσμο μας καθώς και τα βασικά αντικείμενα και χαρακτηριστικά του μικρόκοσμου. Μερικοί από τους κανόνες που θέσαμε είναι οι εξής:

- θεωρούμε ότι οι αγώνες έχουν διάρκεια 90 λεπτά, δεν έχουμε καθυστερήσεις και ότι σε κάθε αγώνα συμμετέχουν δύο ομάδες,
- σε κάθε αγώνα, υπάρχουν 4 διαιτητές με διαφορετικές ιδιότητες ο καθένας. Οι ιδιότητες είναι αυτή του Πρώτου Διαιτητή, του Πρώτου και Δεύτερου Βοηθού διαιτητή, καθώς και του 4^{ου} Διαιτητή. Κάθε διαιτητής μπορεί να συμμετέχει από διαφορετικό πόστο σε κάθε αγώνα, δηλαδή δεν υπάρχουν στάνταρ πρώτοι διαιτητές, τέταρτοι διαιτητές και βοηθοί διαιτητών,
- σε κάθε αγώνα, μία ομάδα παρατάσσει 11 βασικούς παίκτες κάθε φορά, ενώ ακόμη έχει ορισμένο αριθμό αναπληρωματικών παίκτων από τους οποίους μπορούν να μπουν μέχρι και 4 ως αλλαγή με κάποιον άλλο παίκτη που ήδη συμμετείχε στον αγώνα,
- στο πρωτάθλημα αυτό θεωρήσαμε ότι η βαθμολογία για κάθε αγώνα μετριέται ως εξής: 3 βαθμούς για νίκη, 1 βαθμό για ισοπαλία και 0 βαθμούς για ήττα,
- κάθε ομάδα έχει έναν προπονητή, ο οποίος λειτουργεί σαν Coach-Manager της ομάδας (δεν υπάρχει δηλαδή κάποιο επιπλέον προπονητικό-team),

- ακόμη, θεωρούμε ότι μία ομάδα μπορεί να κάνει μεταγραφές παικτών καθ' όλη τη διάρκεια της σεζόν,
- κάθε Στάδιο/Γήπεδο μπορεί να είναι έδρα για πολλές ομάδες του πρωταθλήματος ενώ μπορεί ακόμη να στελεχώσει πολλούς αγώνες καθ' όλη τη διάρκεια της σεζόν,
- ο Πρώτος διαιτητής μπορεί να καταλογίσει παράβαση σε έναν παίκτη κατά τη διάρκεια ενός αγώνα, και αν το κρίνει αναγκαίο, να του δώσει μία κάρτα. Η κάρτα αυτή μπορεί να είναι κίτρινη και κόκκινη. Σε περίπτωση κίτρινης κάρτας έχουμε απλή προειδοποίηση, ενώ σε περίπτωση δεύτερης κίτρινης κάρτας στον ίδιο αγώνα ή κόκκινης κάρτας, έχουμε αποβολή παίκτη.

Έχοντας ορίσει τους παραπάνω κανόνες, προχωρήσαμε στη δημιουργία του ERD το οποίο ικανοποιεί τους κανόνες και περιορισμούς αυτούς και το οποίο θα αναλυθεί αναλυτικά παρακάτω.

Αξιολόγηση:

Μερικούς από τους τρόπους με τους οποίους αξιολογήσαμε το πρότζεκτ ήταν οι ακόλουθοι:

- να είναι εύκολη στη χρήση η εφαρμογή,
- να μην δημιουργούνται προβλήματα όσον αφορά τη λειτουργία της εφαρμογής μας,
- τα βασικά δεδομένα που βάλαμε να αντικατοπτρίζουν την πραγματικότητα. Δεδομένου ότι ορισμένα από τα δεδομένα τα πήραμε από ιστοσελίδες μέσω Web Scraping, περιμέναμε να αντιστοιχούν με τα ανάλογα δεδομένα των ιστοσελίδων αυτών, όπως πχ. ο βαθμολογικός πίνακας, οι αγώνες, οι παίκτες, τα γκολ κάθε ομάδας σε έναν αγώνα, οι προπονητές, ο πρώτος σκόρερ και άλλα,
- να μπορούν να υλοποιούνται και να είναι και σχετικά γρήγορα τα «ερωτήματα» που μπορεί να θέσει ο χρήστης σχετικά με το πρωτάθλημα (βαθμολογικός πίνακας, αγώνες, μερικά ενδιαφέροντα στατιστικά όπως κορυφαίος σκόρερ και άλλα),
- να ικανοποιούνται οι περιορισμοί που θέσαμε κατά τη διάρκεια του πρότζεκτ.

Τα κριτήρια επιτυχίας που χρησιμοποιήσαμε ήταν τα εξής:

- αρχικά, θέλαμε να πετύχουμε το επιδιωκόμενο αποτέλεσμα για το πρότζεκτ με τη βέλτιστη ποιότητα (χρήση γραφικής διεπαφής),
- θέλαμε να μένουμε πιστοί στις προθεσμίες που θέταμε για κάθε μέρος του πρότζεκτ,
- θέλαμε εμείς οι ίδιοι να είμαστε ικανοποιημένοι με αυτό που φτιάχνουμε,
- ακόμη, θέλαμε και ο χρήστης της εφαρμογής να μένει ικανοποιημένος από αυτή και να μπορεί να βλέπει τις πληροφορίες που πιθανώς θα ήθελε να δει χωρίς κανένα πρόβλημα.

Δεδομένα:

Προκειμένου να αξιολογήσουμε αν η βάση που δημιουργήσαμε λειτουργεί σωστά, συλλέξαμε δεδομένα από το πρωτάθλημα της Premier League 2020-2021, διότι σε αυτό υπήρχαν πολλά δεδομένα online τα οποία χρειαζόμασταν για τη βάση μας. Τα site που χρησιμοποιήσαμε για τη συλλογή των δεδομένων είναι τα εξής: <https://fbref.com>^[1], https://en.wikipedia.org/wiki/List_of_Premier_League_managers^[2], <https://www.worldfootball.net>^[3] και <https://www.premierleague.com>^[4]. Για να πάρουμε τα ακριβή δεδομένα που χρειαζόμασταν χρησιμοποιήσαμε τη μέθοδο web scraping, η οποία χρησιμοποιείται για την απόκτηση και συλλογή δεδομένων από το διαδίκτυο. Αυτή η μέθοδος υλοποιήθηκε μέσω της Python με χρήση των βιβλιοθηκών BeautifulSoup και requests. Έπειτα για την ενσωμάτωση των δεδομένων στη βάση μας φτιάχναμε αρχεία excel και csv μέσω της Python με χρήση των βιβλιοθηκών pandas, xlsxwriter. Στη συνέχεια δημιουργήσαμε μια σύνδεση στο πρόγραμμα μας με τη βάση δεδομένων και μέσω της βιβλιοθήκης pandas, στην οποία φορτώσαμε τα δεδομένα από τα excel/csv files, τα μεταφέραμε στη βάση δεδομένων μας.

Ενέργειες ολοκλήρωσης πρότζεκτ:

Αρχικά σκεφτήκαμε τους περιορισμούς και τις παραδοχές για τον μικρόκοσμο μας και δημιουργήσαμε το ERD (Entity Relation Diagram) (<https://erdmaker.com>) το οποίο ήταν η βάση του πρότζεκτ μας. Εν συνεχεία μετατρέψαμε το ERD μας στο αντίστοιχο σχεσιακό σχήμα (<https://www.dbdesigner.net>) και ορίσαμε τον τύπο δεδομένων των γνωρισμάτων της κάθε σχέσης. Έπειτα δημιουργήσαμε τη βάση δεδομένων μας μέσω Python, την οποία συνδέσαμε με την SQLite και μέσω κάποιων queries (create) δημιουργήσαμε τους πίνακες, τα γνωρίσματα τους, καθώς και τους περιορισμούς τους. Στη συνέχεια με τη χρήση του web scraping συλλέξαμε τα δεδομένα που ήταν απαραίτητα για τη βάση μας και τα εντάξαμε σε csv/excel files και μέσω κώδικα τα ενσωματώσαμε στους αντίστοιχους πίνακες. Τέλος δημιουργήσαμε την εφαρμογή που διαθέτει γραφική διεπαφή, στην οποία εμφανίζονται αποτελέσματα τυπικών αναζητήσεων πιο παραστατικά, μέσω της οποίας επίσης μπορούμε να τροποποιήσουμε κάποια από τα ήδη υπάρχοντα δεδομένα. Επιπλέον συντάξαμε την αναφορά του πρότζεκτ και δημιουργήσαμε την παρουσίαση του και δημιουργήσαμε ένα file με τις οδηγίες εγκατάστασης.

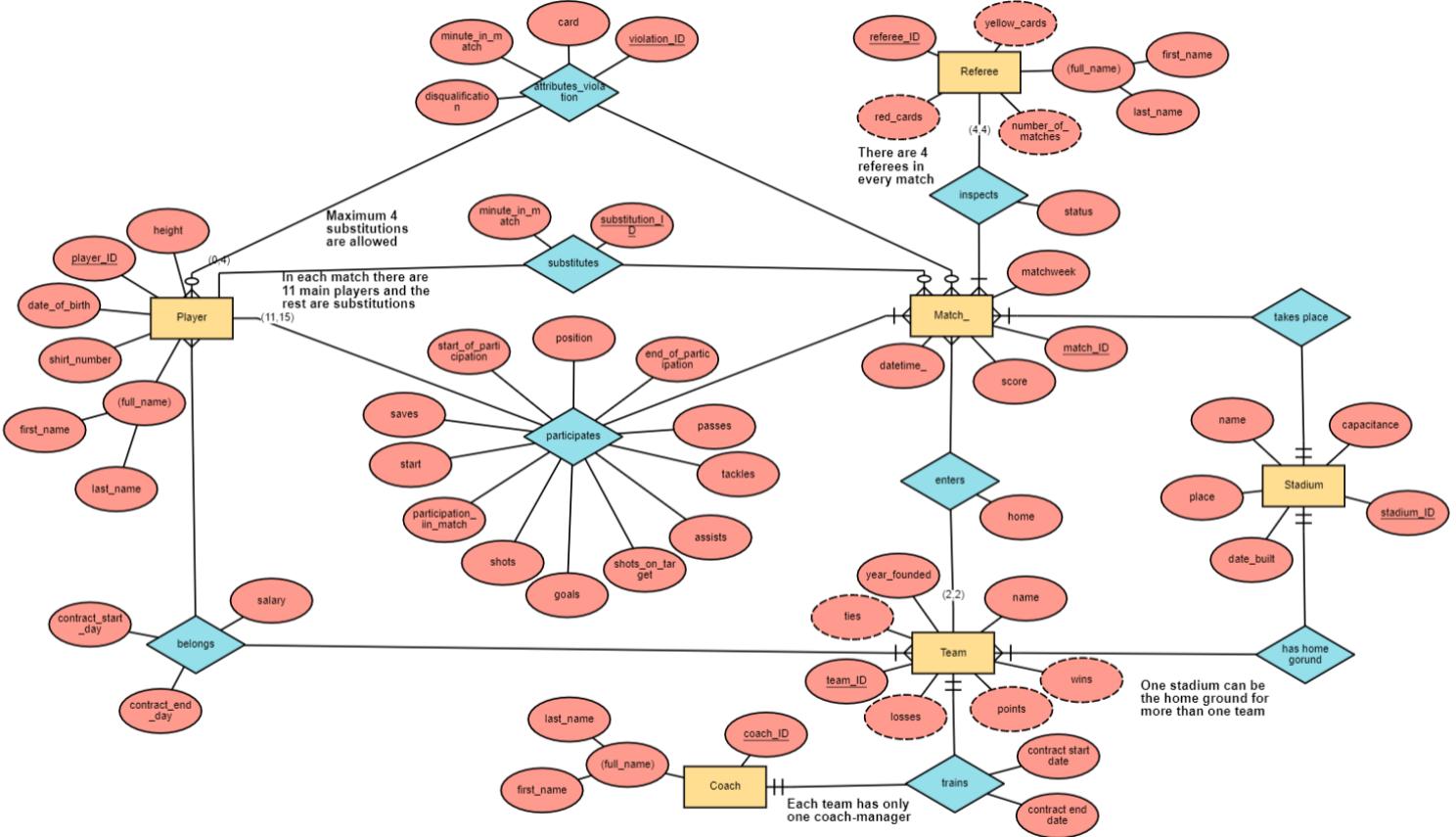
Αρμοδιότητες ομάδας:

Αρχικά για το ERD κάναμε τακτικές συναντήσεις μέχρι να καταλήξουμε στο τελικό διάγραμμα, οπότε έγινε ομαδικά, όπως και η μετατροπή στο αντίστοιχο σχεσιακό σχήμα και η δημιουργία της βάσης δεδομένων. Έπειτα για τη συλλογή των δεδομένων με χρήση web scraping, λόγω του μεγάλου όγκου δεδομένων που είχαμε να συλλέξουμε, ανέλαβε ο καθένας το μάζεμα των δεδομένων για ένα μερίδιο των πινάκων. Συγκεκριμένα ο Ιωάννης ανέλαβε τους πίνακες: Match_, enters, substitutes, Referee, inspects και attributesViolation, ενώ η Κωνσταντίνα ανέλαβε τους πίνακες: Player, participates, Team, Stadium, Coach και belongs. Επιπροσθέτως, για τη δημιουργία της εφαρμογής ο Ιωάννης ανέλαβε τη δημιουργία της αρχικής σελίδας της εφαρμογής, ενώ η Κωνσταντίνα ανέλαβε τη σχεδίαση των παραθύρων και τη μορφοποίηση τους, προκειμένου να είναι καλύτερα διαμορφωμένα τα πεδία της εφαρμογής, καθώς και τη δημιουργία της σελίδας που προκύπτει όταν ο χρήστης πατήσει “more stats” στην αρχική σελίδα. Τέλος η σύνταξη της αναφοράς, της παρουσίασης και της δημιουργίας των οδηγιών εγκατάστασης πραγματοποιήθηκαν έπειτα από διάφορες συναντήσεις, τις οποίες κάναμε μαζί.

Χρονοδιάγραμμα:

- ERD: 27/10 (μόλις ανακοινώθηκαν τα θέματα) – 29/11
- Σχεσιακό σχήμα: 20/11-29/11 (29/11 ήταν η τελική έκδοση του ERD οπότε έγιναν κάποιες αλλαγές και στο σχεσιακό σχήμα)
- Δημιουργία βάσης: 30/11
- Συλλογή δεδομένων για τη βάση: 1/12-26/12 (ήταν μεγάλος ο όγκος δεδομένων και συναντήσαμε αρκετά προβλήματα)
- Σύνταξη εντολών SQL για τροποποίηση της βάσης και για τυπικές αναζητήσεις: 26/12-29/12
- Δημιουργία εφαρμογής: 30/12-3/1
- Σύνταξη αναφοράς: 3/1-14/1
- Δημιουργία παρουσίασης: 15/1-16/1
- Δημιουργία οδηγιών εγκατάστασης: 16/1

ERD:



Εικόνα 1: Entity Relationship Diagram (ERD)

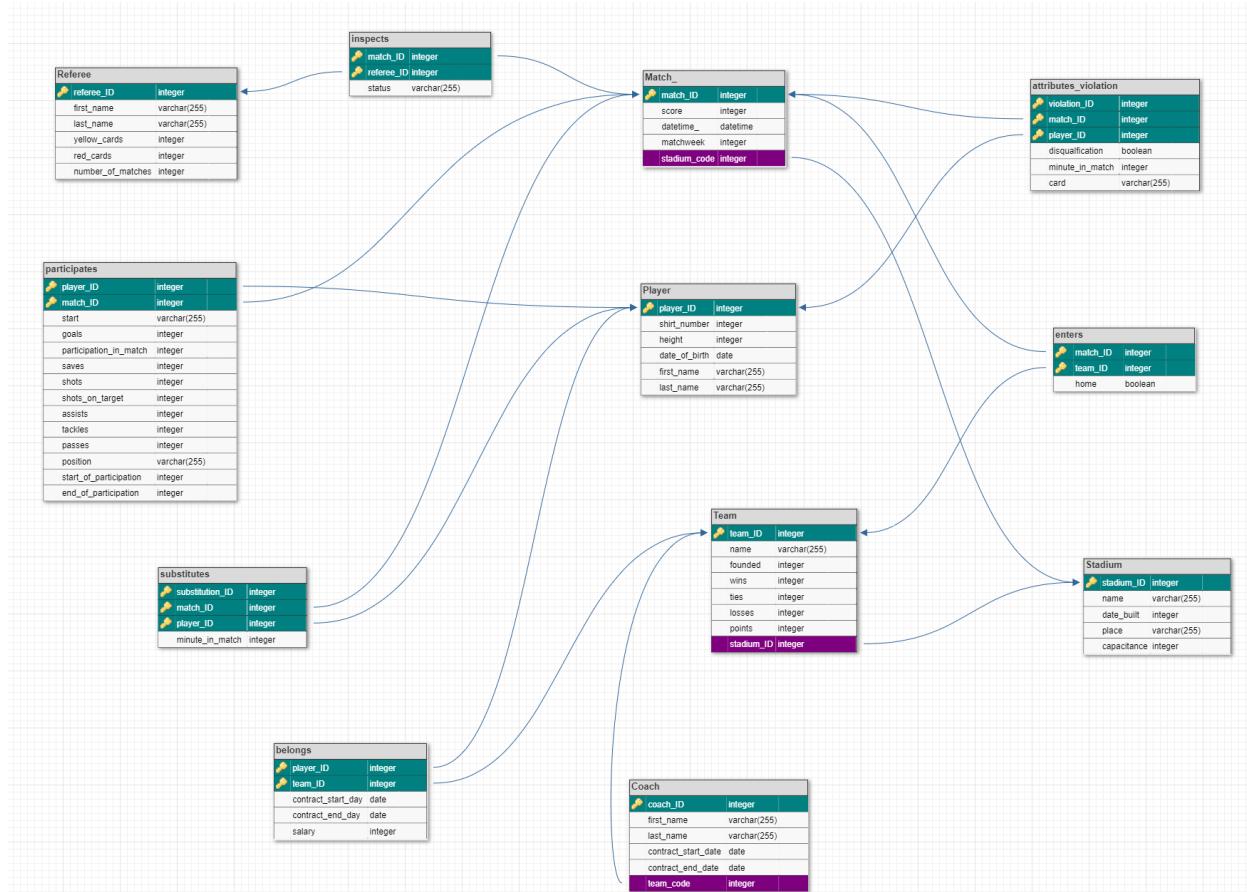
Περιγραφή του ERD:

Πριν ξεκινήσουμε το ERD θέσαμε κάποιους περιορισμούς, τους οποίους αναλόγουμε στη **Μεθοδολογία** (σελίδα 1 κάτω μέρος - σελίδα 2 πάνω μέρος), στους οποίους βασίστηκε το διάγραμμά μας. Όσον αφορά το ERD, αρχικά έχουμε την οντότητα **Team**, η οποία περιλαμβάνει τις ομάδες που συμμετέχουν στο τοπικό πρωτάθλημα ποδοσφαίρου, η κάθε ομάδα έχει σαν έδρα ένα γήπεδο, γι' αυτό δημιουργήσαμε και την οντότητα **Stadium**, για να καταγράψουμε δηλαδή τα γήπεδα που είναι έδρες των ομάδων που συμμετέχουν στο πρωτάθλημα. Το κάθε γήπεδο συσχετίζεται με την ομάδα μέσω της συσχέτισης **has home ground** και επειδή ασχοληθήκαμε με τοπικό πρωτάθλημα μπορεί να φιλοξενεί παραπάνω από μια ομάδες, εξών και η πληθικότητα που βάλαμε. Επιπλέον κάθε ομάδα έχει τον δικό της προπονητή οπότε δημιουργήσαμε την οντότητα **Coach**. Ο προπονητής λειτουργεί και σαν μάνατζερ της ομάδας και δεν έχει βιοθούς, γι' αυτό υπάρχει μοναδικός προπονητής για κάθε ομάδα. Ο προπονητής συνδέεται με την ομάδα μέσω της συσχέτισης **trains**. Επιπροσθέτως, κάθε ομάδα αποτελείται από κάποιους παίκτες, οπότε δημιουργήσαμε την οντότητα **Player** στην οποία θα καταγράψουμε τους παίκτες που είναι μέλη κάποιας εκ των ομάδων που συμμετέχουν στο πρωτάθλημα. Η συσχέτιση που συνδέει τον **Player** με την **Team** είναι η **belongs** στην οποία καταγράφουμε τα συμβόλαια και τους μισθούς των παίκτων. Επίσης επιλέξαμε οι παίκτες να ανήκουν σε μια ή περισσότερες ομάδες, ώστε να μπορούμε να καταγράψουμε και τις τυχόν μεταγραφές που έγιναν κατά τη διάρκεια του πρωταθλήματος. Ακόμη, στο τοπικό πρωτάθλημα οι ομάδες ανά δυο συμμετέχουν σε αγώνες, οπότε δημιουργήσαμε την οντότητα **Match_** για να καταγράψουμε όλους τους αγώνες κατά τη διάρκεια του πρωταθλήματος και να γνωρίζουμε την αγωνιστική, την ημερομηνία που έγιναν, το σκορ κ.ά. Συσχετίζουμε τους αγώνες και τις ομάδες μέσω της συσχέτισης **enters**, η οποία δηλώνει και αν η ομάδα που συμμετείχε στον αγώνα είναι γηπεδούχος ή όχι μέσω του Boolean γνωρίσματος **home**. Κάθε αγώνας λαμβάνει χώρα σε ένα από τα γήπεδα της βάσης

μας και γι' αυτό συνδέονται με τη συσχέτιση `takes place` και διαθέτει τέσσερις διαιτητές, οπότε δημιουργήσαμε και την οντότητα `Referee`, στην οποία καταγράφουμε τους διαιτητές που διαιτήσαν αγώνες του πρωταθλήματος. Οι διαιτητές συνδέονται με τους αγώνες μέσω της συσχέτισης `inspects` στην οποία καταγράφουμε και τη θέση του διαιτητή, δηλαδή αν ήταν πρώτος διαιτητής, τέταρτος διαιτητής ή ένας εκ των δύο βοηθών διαιτητή. Σε κάθε αγώνα μιας ομάδας συμμετέχουν από έντεκα έως και το πολύ δεκαπέντε παίκτες (11 βασικοί και 4 το πολύ αλλαγές), που έπειτα από τη συμμετοχή τους προκύπτουν τα στατιστικά τους για τον συγκεκριμένο αγώνα. Συνδέουμε τις οντότητες του παίκτη και του αγώνα με τη συσχέτιση `participates`, στην οποία καταγράφονται τα στατιστικά του παίκτη που συμμετείχε και άλλα στοιχεία, όπως αν ήταν από τους βασικούς, σε ποια θέση έπαιξε και τα λεπτά που συμμετείχε. Προκειμένου να καταγράψουμε τις αλλαγές που γίνονται κατά τη διάρκεια ενός αγώνα δημιουργήσαμε τη συσχέτιση `substitutes`, η οποία ενώνει τον παίκτη και τον αγώνα. Σε αυτή καταγράφεται το λεπτό της αλλαγής, καθώς και ένα κλειδί για την κάθε αλλαγή. Επιλέξαμε να βάλουμε σαν γνώρισμα αυτό το κλειδί προκειμένου να γνωρίζουμε ποιους παίκτες αφορά η κάθε αλλαγή και να βεβαιωθούμε ότι αν κάποιος μπήκε ως αλλαγή και για κάποιο λόγο ξαναβγήκε να μην εμφανιστεί το ίδιο πρωτεύον κλειδί δεύτερη φορά. Προκειμένου να καταγράψουμε τις παραβάσεις που οδηγούν σε κάρτα και γίνονται κατά τη διάρκεια ενός αγώνα δημιουργήσαμε τη συσχέτιση `attributesViolation`, η οποία ενώνει τον παίκτη και τον αγώνα. Σε αυτή καταγράφεται το λεπτό της παραβάσης, η κάρτα που του δόθηκε και αν αποκλείστηκε από τον αγώνα, καθώς και ένα κλειδί για την κάθε παραβάση. Επιλέξαμε να βάλουμε σαν γνώρισμα αυτό το κλειδί προκειμένου να μην εμφανιστεί το ίδιο πρωτεύον κλειδί δεύτερη φορά αν κάποιος παίκτης πάρει δεύτερη κάρτα (δεύτερη κίτρινη ή κόκκινη κάρτα) στον ίδιο αγώνα. Ουσιαστικά μέσα από αυτό το διάγραμμα επιδιώκουμε να μπορούμε να εμφανίσουμε τους παίκτες των ομάδων, τις ομάδες, την κατάταξη, καθώς και πιο σύνθετα ερωτήματα που μπορεί να έχει κάποιος χρήστης.

Relational Schema:

Αφού δημιουργήσαμε το ERD έπειτα προχωρήσαμε στη μετατροπή του σε Relational Schema. Η μετατροπή αυτή έγινε βασιζόμενη στη θεωρία και το Relational Schema φαίνεται παρακάτω:



Εικόνα 2: Relational Schema

Συνολικά δημιουργούνται δώδεκα πίνακες, έξι εξ αυτών προέκυψαν από τις ισχυρές οντότητες και οι υπόλοιποι έξι από συσχετίσεις (πολλά προς πολλά).

Web Scraping:

Προκειμένου να συλλέξουμε τα δεδομένα του τοπικού πρωταθλήματος ποδοσφαίρου επιλέξαμε να χρησιμοποιήσουμε πραγματικά δεδομένα από το πρωτάθλημα της Premier League 2020-2021. Η συλλογή των δεδομένων έγινε με τη μέθοδο του Web Scraping μέσω της Python με χρήση των βιβλιοθηκών BeautifulSoup και Requests. Κάνοντας αυτή τη διαδικασία δημιουργούμε excel/csv αρχεία, τα οποία στη συνέχεια τα εισάγουμε στη βάση μας μέσω της Python με χρήση της βιβλιοθήκης Pandas. Κάποια παραδείγματα προγραμμάτων φαίνονται παρακάτω:

Συλλογή δεδομένων για τους προπονητές-μάνατζερ των ομάδων:

```
coaches.py - C:\Users\konst\Desktop\Υο εξαμηνό\project\data collection codes\coaches.py (3.10.1)
File Edit Format Run Options Window Help
from bs4 import BeautifulSoup
import requests
import xlsxwriter as xx

#setting the headers in the .xlsx file
headers = ['coach_ID','first_name','last_name','contract_start_date','contract_end_date','team_code']

#create the .xlsx file
workbook = xx.Workbook('Coach.xlsx')
worksheet = workbook.add_worksheet()

row=0
col=0
#writing the name of the headers in the first row of the .xlsx file
for count,l in enumerate(headers):
    worksheet.write(row, count,l)
row=row+1

#the list of teams to make it easier to find the team code
list_teams=["Manchester City","Manchester United","Liverpool","Chelsea","Leicester City","West Ham United",\
    "Tottenham Hotspur","Arsenal","Leeds United","Everton","Aston Villa","Newcastle United",\
    "Wolverhampton Wanderers","Crystal Palace","Southampton","Brighton & Hove Albion",\
    "Burnley","Fulham","West Bromwich Albion","Sheffield United"]#teams of premier league 2020-2021

coach_id=0
#getting the html code from the site that contains the coaches of the teams
link="https://en.wikipedia.org/wiki/List_of_Premier_League_managers"
html_text = requests.get(link).text
coaches_soup = BeautifulSoup(html_text,'lxml')
coaches = coaches_soup.find_all("tbody")[1].find_all("tr")#contains all the rows of the table
for count,coach in enumerate(coaches):
    if(count==0):#the first row were the headers
        continue
    team=coach.find_all("td")[1].text.strip()#team of the current coach
    if team in list_teams: #if he is a coach of a team that participated in Premier League 2020-2021
        general=coach.find_all("td")
        try:
            #we wanted to gather the coaches that were active during the Premier League 2020-2021
            #so if his contract ended before the start of the championship he wasn't included or if the contract
            #started after the end of the championship
            if(int(general[2].span.get('data-sort-value'))[13:15])>5 and int(general[2].span.get('data-sort-value'))[8:12]==2021:
                continue #not accepted
            if(int(general[3].span.get('data-sort-value'))[8:12])>=2020:
                if(int(general[3].span.get('data-sort-value'))[8:12])==2020 and int(general[3].span.get('data-sort-value'))[13:15]<9:#not accepted
                    continue
                name=coach.th.text.split(" ") #full name of the coach
                fname=name[0].strip() #first name of the coach
                lname=name[1].strip() #last name of the coach
                contract_end=general[3].span.get('data-sort-value')[8:18] #date the contract of the coach ended
                if(contract_end[0:4]=='2021' and (contract_end[5:7]=='05' or contract_end[5:7]=='06')):
                    contract_end='2021-06-30' #there were some gaps in the end of the championship so we extended their contract
                contract_start=general[2].span.get('data-sort-value')[8:18] #date the contract of the coach began
                try:
                    team_code=list_teams.index(general[1].text.strip()) #team code he was the coach of
                except:
                    team_code='NULL'
                #writing the information we gathered to the .xlsx file
                worksheet.write(row,col,coach_id)
                worksheet.write(row,col+1,fname)
                worksheet.write(row,col+2,lname)
                worksheet.write(row,col+3,contract_start)
                worksheet.write(row,col+4,contract_end)
                worksheet.write(row,col+5,team_code)
                coach_id=coach_id+1
                row=row+1
            except:#accepted because he remains a coach to this date (the contract end was null)
                try:
                    team_code=list_teams.index(general[1].text.strip()) #team code he was the coach of
                except:
                    team_code='NULL'
                name=coach.th.text.split(" ") #full name of the coach
                fname=name[0].strip() #first name of the coach
                lname=name[1].strip() #last name of the coach
                contract_end='NULL' #he's still coach for this team
                contract_start=general[2].span.get('data-sort-value')[8:18] #date the contract of the coach began
                #writing the information we gathered to the .xlsx file
                worksheet.write(row,col,coach_id)
                worksheet.write(row,col+1,fname)
                worksheet.write(row,col+2,lname)
                worksheet.write(row,col+3,contract_start)
                worksheet.write(row,col+4,contract_end)
                worksheet.write(row,col+5,team_code)
                coach_id=coach_id+1
                row=row+1
        workbook.close()
```

Εικόνα 3: Web Scraping για τον πίνακα Coach

Ανάλυση προγράμματος:

Αρχικά κάνουμε import τις βιβλιοθήκες BeautifulSoup, xlsxwriter, requests. Έπειτα δημιουργούμε ένα excel workbook για να περάσουμε εκεί τα δεδομένα από το Web scraping και στην πρώτη του γραμμή γράφουμε τα ονόματα των γνωρισμάτων του πίνακα Coach. Στη συνέχεια δημιουργούμε μια λίστα με τα ονόματα των ομάδων (με τέτοια σειρά ώστε το index τους να δίνει το team_ID τους στον πίνακα Team), ώστε για κάθε προπονητή να βρίσκουμε το team code της ομάδας στην οποία ήταν προπονητής. Έπειτα ξεκινάμε το Web Scraping από τη σελίδα https://en.wikipedia.org/wiki/List_of_Premier_League_managers που υπάρχουν τα συμβόλαια των προπονητών. Αρχικά με τη χρήση της requests και της BeautifulSoup παίρνουμε τον κώδικα html της σελίδας και βρίσκουμε αρχικά το tbody που περιέχει τον πίνακα με τους προπονητές και στη συνέχεια βρίσκοντας όλα τα τι παίρνουμε τις γραμμές αυτού του πίνακα (κάθε τι αντιστοιχεί σε μια γραμμή του πίνακα). Οπότε παίρνοντας μια-μια τις γραμμές (με εξαίρεση την πρώτη που είχε τους headers) βρίσκαμε όλα τα τι κάθε γραμμής, όπου το καθένα περιείχε διαφορετικό στοιχείο για τον προπονητή, όπως τις ημερομηνίες έναρξης και λήξης συμβολαίου. Έπειτα θέλαμε να φιλτράρουμε τους προπονητές που το συμβόλαιο τους έληξε πριν την έναρξη του πρωταθλήματος και αυτούς που το συμβόλαιο τους ξεκίνησε μετά την λήξη του πρωταθλήματος. Επιπλέον επειδή σε κάποιους το συμβόλαιο έληγε ένα μήνα πριν το τέλος του πρωταθλήματος ή λίγο πριν τη λήξη του επιλέξαμε για χάριν απλότητας να επεκτείνουμε το συμβόλαιο τους μέχρι τις 2021-06-30, δηλαδή αμέσως μετά τη λήξη του πρωταθλήματος. Σε κάποιους δεν υπήρχε λήξη συμβολαίου γιατί παραμένουν προπονητές μέχρι σήμερα, οπότε η ημερομηνία λήξης του συμβολαίου τους ήταν NULL, γι' αυτό προσθέσαμε ένα try, except. Τέλος γράφουμε στις γραμμές του excel τα αποτελέσματα για κάθε προπονητή μέσω της βιβλιοθήκης xlsxwriter.

Συλλογή δεδομένων για τον πίνακα participates:

```
participates.py - C:\Users\konst\Desktop\Το εξαμνο\project\data collection codes\participates.py (3.10.1)
File Edit Format Run Options Window Help
from bs4 import BeautifulSoup
import requests
import random as rd
import xlsxwriter as xw
import pandas as pd

#setting the headers in the .xlsx file
headers = ['player_ID','match_ID','start','goals','participation_in_match','saves','shots','shots_on_target','assists','tackles','passes','position']

#create the .xlsx file
workbook = xw.Workbook('participates.xlsx')
worksheet = workbook.add_worksheet()

row=0
col=0
#writing the name of the headers in the first row of the .xlsx file
for count,l in enumerate(headers):
    worksheet.write(row, count,l)

#an array that contains the names of the teams to make it easier to find the right match ids
t=['Manchester City','Manchester Utd','Liverpool','Chelsea','Leicester City','West Ham','Tottenham','Arsenal','Leeds United',\
    'Everton','Aston Villa','Newcastle Utd','Wolves','Crystal Palace','Southampton','Brighton','Burnley','Fulham','West Brom','Sheffield Utd']
d={}
matches=pd.read_excel("matches2.xlsx")#loading the .xlsx file
df=pd.DataFrame(matches) #df contains the information about the matches and the two teams that participated in the match

player_id=-1
#getting the html code from the site that contains the teams that took part in Premier League 2020-2021
link="https://fbref.com/en/comps/9/10728/2020-2021-Premier-League-Stats"
html_text = requests.get(link).text
teams_soup = BeautifulSoup(html_text,'lxml')
teams = teams_soup.find("tbody").find_all('a')#it contains all the teams
for team in teams:
    part_link=team.get('href').split('/')
    if(part_link[2]=='squads'):
        print("done") #the program is slow so we needed to know if it was getting close to the end
        new_link="https://fbref.com"+"/".join(part_link) #link of the list of players of each team
        #getting the html code of the list of players
        html_text2=requests.get(new_link).text
        players_soup=BeautifulSoup(html_text2,'lxml')
        players1=players_soup.find("tbody")
        players2=players1.find_all("td",{"data-stat":"matches"}) #contains each player of the list
        for count1,player in enumerate(players2):
            player_id=player_id+1
            split=player.a.get('href').split('/')
```

```

position2=players1.find_all("tr")[count1].find("td", {"class": "center"}).text.strip() #the position the player is playing
stats_link="https://fbref.com/en/players/+split[3]+/matchlogs/s10728/summary/+split[-1] #the link of the stats of the player for Premier League 2020-2021
#getting the html code of the stats of the player
html_text3=requests.get(stats_link).text
stats_soup=BeautifulSoup(html_text3,'lxml')
stats=stats_soup.find("tbody").find_all("tr")
for stat in stats: #the stats for each match that he took part
    if(position2!="GR"):
        saves="NULL"#only goalkeepers have "saves"
    else:
        saves=d.randint(4,8)#saves weren't provided in the site
    position=stat.find("td", {"data-stat": "position"})#position for the specific match
    if(position==None):#there were blank lines and we wanted to skip them
        continue
    else:
        venue=stat.find("td", {"data-stat": "venue"}).text #venue of the match to make it easier to find match ids
        if venue=="Home":
            venue=1
        else:
            venue=0
        squad=stat.find("td", {"data-stat": "squad"}).text #team of the player
        opponent=stat.find("td", {"data-stat": "opponent"}).text #the opponent in that match
        d.append([venue,squad,opponent])#helping list for the match ids
        minutes=stat.find("td", {"data-stat": "minutes"}).text.strip()#minutes the player participated in the match
        starterr=stat.find("td", {"data-stat": "game_started"}).text #if he was one of the main players in the match
        goals=stat.find("td", {"data-stat": "goals"}).text #how many goals he scored
        shots=stat.find("td", {"data-stat": "shots_total"}).text #how many shots
        shots_on_target=stat.find("td", {"data-stat": "shots_on_target"}).text #how many shots on target
        assists=stat.find("td", {"data-stat": "assists"}).text #how many assists
        tackles=stat.find("td", {"data-stat": "tackles"}).text #how many tackles
        passes=stat.find("td", {"data-stat": "passes_completed"}).text #how many passes
    row=row+1
#writing the information we gathered to the .xlsx file
worksheet.write(row,col,player_id)
worksheet.write(row,col+2,starterr)
worksheet.write(row,col+3,goals)
worksheet.write(row,col+4,minutes)
worksheet.write(row,col+5,saves)
worksheet.write(row,col+6,shots)
worksheet.write(row,col+7,shots_on_target)
worksheet.write(row,col+8,assists)
worksheet.write(row,col+9,tackles)
worksheet.write(row,col+10,passes)
worksheet.write(row,col+11,position2)

row=0
#finding the correct ids for each match and writing them to the .xlsx file
for x in range(0,len(d)):
    for y in range(0,len(df)):
        if d[x][0]==1 and df.at[y, 'squad_a']==d[x][1] and df.at[y, 'squad_b']==d[x][2]:
            match_ID=df.at[y, 'match_ID']
        elif d[x][0]==0 and df.at[y, 'squad_b']==d[x][1] and df.at[y, 'squad_a']==d[x][2]:
            match_ID=df.at[y, 'match_ID']
    row+=1
    worksheet.write(row,col+1,match_ID)
workbook.close()

```

Εικόνα 4: Web Scraping για τον πίνακα participates

Ανάλυση προγράμματος:

Αρχικά κάνουμε import τις βιβλιοθήκες BeautifulSoup, xlsxwriter, requests, pandas. Έπειτα δημιουργούμε ένα excel workbook για να περάσουμε εκεί τα δεδομένα από το Web scraping και στην πρώτη του γραμμή γράφουμε τα ονόματα των γνωρισμάτων του πίνακα participates. Στη συνέχεια δημιουργούμε μια λίστα με τα ονόματα των ομάδων (με τέτοια σειρά ώστε το index τους να δίνει το team_ID τους στον πίνακα Team), ώστε να βρούμε πιο εύκολα το match_ID. Επιπλέον με χρήση της βιβλιοθήκης pandas φορτώσαμε το excel matches2, το οποίο περιέχει τα match και το χρειαζόμαστε για τον υπολογισμό των match_ID. Έπειτα ξεκινάμε το Web Scraping από τη σελίδα της <https://fbref.com/en/comps/9/10728/2020-2021-Premier-League-Stats> που υπήρχε η λίστα των ομάδων. Αρχικά με τη χρήση της requests και της BeautifulSoup πάρινουμε τον κώδικα html της σελίδας και βρίσκουμε αρχικά το tbody που περιέχει τον πίνακα με τις ομάδες που πήραν μέρος στο πρωτάθλημα της Premier League 2020-2021. Έπειτα βρίσκουμε όλα τα αγαθά μέσα σε αυτά περιέχεται το href που μας δίνει το μέρος του link που χρειαζόμαστε ώστε για κάθε ομάδα να μπούμε στο link της και να βρούμε τους παίκτες της. Έπειτα με τη χρήση της requests και της BeautifulSoup πάρινουμε τον κώδικα html της νέας σελίδας που περιέχει τα ονόματα των παικτών της ομάδας και βρίσκουμε αρχικά το tbody που περιέχει τον πίνακα με τα ονόματα των παικτών. Από το tbody βρίσκω όλα τα "td", {"data-stat": "matches"}, διότι αυτά περιέχουν τις σειρές του πίνακα (κάθε σειρά του πίνακα αφορά έναν παίκτη). Έπειτα για κάθε παίκτη θα πάρω το link που με οδηγεί στα match reports του. Εκεί με τη χρήση της requests και της BeautifulSoup πάρινουμε τον κώδικα html της νέας σελίδας που περιέχει στατιστικά του παίκτη για κάθε αγώνα στον οποίο συμμετείχε στην Premier League 2020-2021 και βρίσκουμε αρχικά το tbody που περιέχει τον πίνακα με τα στατιστικά του και στη συνέχεια βρίσκοντας όλα τα tr παίρνουμε τις γραμμές αυτού του πίνακα (κάθε tr αντιστοιχεί σε μια γραμμή του πίνακα που περιέχει στατιστικά για ένα συγκεκριμένο αγώνα). Οπότε παίρνοντας μια-μια τις γραμμές (με εξαίρεση τις κενές γραμμές που υπάρχουν) βρίσκαμε τα td κάθε γραμμής που αντιστοιχούν στα γνωρισμάτα που θέλουμε να αποθηκεύσουμε στο excel μας και κατ' επέκτασιν στη βάση μας. Επιπλέον κρατάμε μια λίστα με την

ομάδα, την αντίπαλο και ποια ήταν γηπεδούχος. Τέλος γράφουμε στις γραμμές του excel τα αποτελέσματα για κάθε συμμετοχή παίκτη σε αγώνα μέσω της βιβλιοθήκης xlswriter. Έπειτα για να βρούμε τα match_ID συγκρίνουμε την ομάδα, τον αντίπαλο και το ποια ήταν γηπεδούχος με τα στοιχεία που φορτώσαμε από το matches2.xlsx και έπειτα προσθέτουμε τα match_ID στο excel που είχαμε ήδη φτιάξει.

Συλλογή Δεδομένων των Αγώνων του Πρωταθλήματος:

Αρχικά, φορτώνουμε τις βιβλιοθήκες BeautifulSoup, requests, pandas και xlswriter της python, προκειμένου να κάνουμε Web-Scraping τα δεδομένα των αγώνων από την ιστοσελίδα <https://fbref.com/en/comps/9/10728/schedule/2020-2021-Premier-League-Scores-and-Fixtures> καθώς και να βάλουμε τα δεδομένα σε ένα αρχείο excel. Ακόμη, δημιουργούμε μία λίστα με τα ονόματα των σταδίων των ομάδων, τα οποία μπαίνουν με την σειρά με την οποία θα μπουν αυτά στη βάση μας, προκειμένου να ορίσουμε τα στάδια στα οποία θα συμβούν οι αγώνες του πρωταθλήματος (stadium_code) και δημιουργούμε το αρχείο excel των αγώνων με όνομα Matches. Αυτό το αρχείο έχει 5 πεδία, όσα και τα γνωρίσματα του πίνακα Match που φτιάζαμε στο Relational-Schema, τα οποία έχουν το ίδιο όνομα με τα γνωρίσματα της σχέσης Match. Στη συνέχεια, κάνουμε request των html κώδικα της συγκεκριμένης ιστοσελίδας μέσω της συνάρτησης requests.get() και στη συνέχεια μέσω της BeautifulSoup, αποκτάμε πρόσβαση στα δεδομένα της ιστοσελίδας.

```
from bs4 import BeautifulSoup
import requests
import pandas as pd
import xlswriter as xw

#List of Stadiums of teams in the order that they are in our database
#Its going to be used in order to determine the stadium_ID in every match
pitches=[ "The American Express Community Stadium","Anfield","Bramall Lane","Craven Cottage","Elland Road","Emirates Stadium",/
          "Etihad Stadium","Goodison Park","King Power Stadium","London Stadium","Molineux Stadium","Old Trafford","St. Mary's Stadium",/
          "Selhurst Park","St. James' Park","Stamford Bridge","The Hawthorns","Tottenham Hotspur Stadium","Turf Moor","Villa Park"]

#Creation of the excel file of matches
headers = ['match_ID','score','datetime_','matchweek','stadium_code']
workbook = xw.Workbook('matches.xlsx')
worksheet = workbook.add_worksheet()
row=0
col=0
for count,l in enumerate(headers):
    worksheet.write(row, count,l)

#Requesting and inserting data in the excel file using BeautifulSoup
print("##For matches")
#Requesting the html text of the link that's inside the the requests.get() function
html_text = requests.get('https://fbref.com/en/comps/9/10728/schedule/2020-2021-Premier-League-Scores-and-Fixtures')
soup = BeautifulSoup(html_text.text,'html.parser')
#Searching in the html code the table that contains the matches of the Football Championship
table=soup.find('tbody')
#for every row of the table that contains data, we get from every match the score,the time and the date of the match,the matchday and the venue
#and we insert them in the excel file.
matches=table.find_all('tr')
i=0
```

Εικόνα 5: Web Scraping για τον πίνακα matches-μέρος Α

Από κάθε γραμμή του πίνακα των αγώνων του πρωταθλήματος, παίρνουμε κάθε φορά το σκορ του αγώνα, την ημερομηνία και ώρα του αγώνα (τα οποία και συνενώνουμε στη μεταβλητή datetime), την αγωνιστική και το γήπεδο στο οποίο συμβαίνει ο κάθε αγώνας, το οποίο και τελικά μετατρέπουμε στο αντίστοιχο κλειδί του πίνακα Stadiums μέσω της λίστας των αγώνων που φτιάξαμε παραπάνω και τα αποθηκεύουμε σε διαδοχικές στήλες μέσα στο excel. Πριν από την αποθήκευση αυτών, ορίζουμε κάθε φορά και το κλειδί του κάθε αγώνα μέσω της μεταβλητής `i`. Τέλος, αφού ολοκληρώσουμε τη συλλογή των δεδομένων, για να αποθηκεύσουμε το excel χρησιμοποιούμε την εντολή `workbook.close()`.

```

for match in matches:
    fixture=match.find_all('td')
    if(fixture[5].text!=""):
        row+=1
        score=fixture[5].text.strip()
        time=fixture[2].text.strip()
        date=fixture[1].text.strip()
        datetime=date+" "+time
        matchday=match.find('th',{"data-stat":"gameweek"}).text
        venue=pitches.index(fixture[9].text.strip())
        worksheet.write(row,col,i)#This is the ID in which we save the match in our database
        worksheet.write(row,col+1,score)
        worksheet.write(row,col+2,datetime)
        worksheet.write(row,col+3,matchday)
        worksheet.write(row,col+4,venue)
        i+=1
    else:
        continue
#We finish creating the excel file.
workbook.close()

```

Εικόνα 6: Web Scraping για τον πίνακα matches-μέρος Β

Στη συνέχεια, δημιουργούμε ένα δεύτερο αρχείο excel με όνομα `matches2`, το οποίο κάνει την ίδια ακριβώς διαδικασία με παραπάνω, με τη μόνη διαφορά, ότι αποθηκεύει ακόμη τις ομάδες που συμμετέχουν στον αγώνα καθώς και ιστοσελίδες που οδηγούν στα στατιστικά του κάθε αγώνα του πρωταθλήματος. Αυτό το αρχείο χρησιμοποιείται προκειμένου να μπορούμε μέσα από να βγάλουμε εύκολα τη σχέση `enters` καθώς και να μπορούμε να κάνουμε μέσα από αυτό `request` των δεδομένων των αγώνων του πρωταθλήματος (χρησιμοποιώντας τα επιπλέον `links` που βάλαμε στην τελευταία στήλη), χωρίς να χρειάζεται κάθε φορά να κάνουμε Web-scraping για να πάρουμε τις ιστοσελίδες των αγώνων.

```

## We do the same thing as above, but we get also from every match the two teams that play in every match, and the link of match report
headers = ['match_ID','score','datetime','matchday','stadium_code','squad_a','squad_b','Match_Report']
workbook = xw.Workbook('matches2.xlsx')
worksheet = workbook.add_worksheet()
row=0
col=0
for count,l in enumerate(headers):
    worksheet.write(row, count,l)
i=0
for match in matches:
    fixture=match.find_all('td')
    if (fixture[5].text!=""):
        row+=1
        score=fixture[5].text.strip()
        time=fixture[2].text.strip()
        date=fixture[1].text.strip()
        datetime=date+" "+time
        matchday=match.find('th',{"data-stat":"gameweek"}).text
        venue=pitches.index(fixture[9].text.strip())
        squad_a=fixture[3].text.strip()
        squad_b=fixture[7].text.strip()
        link=match.find('td',{"data-stat':'match_report'})
        link=link.find('a')
        link='https://fbref.com'+link['href']
        worksheet.write(row,col,i)
        worksheet.write(row,col+1,score)
        worksheet.write(row,col+2,datetime)
        worksheet.write(row,col+3,matchday)
        worksheet.write(row,col+4,venue)
        worksheet.write(row,col+5,squad_a)
        worksheet.write(row,col+6,squad_b)
        worksheet.write(row,col+7,link)
        i+=1
    else:
        continue
workbook.close()

```

Εικόνα 7: Web Scraping για τον πίνακα matches-μέρος Γ

Διόρθωση χρόνων των πινάκων participates, substitutes, attributes_violation:

Έχοντας τρέξει τα αρχεία participates.py, substitutes.py και attributes_violation.py που βγάζουν τα στατιστικά των παικτών σε κάθε αγώνα που συμμετέχουν, τις αλλαγές και τις κάρτες των παικτών στους διάφορους αγώνες του πρωταθλήματος, τροποποιούμε τους χρόνους σε αυτά προκειμένου να συνολικός χρόνος των παικτών μιας ομάδας σε κάθε αγώνα να είναι το πολύ 990 λεπτά (Το πολύ 990 λεπτά γιατί υπάρχει περίπτωση κάποιος παίκτης να αποβληθεί οπότε ο χρόνος αυτός μειώνεται και δεδομένου ότι ο κάθε αγώνας έχει διάρκεια 90 λεπτά, ο χρόνος αυτός δεν μπορεί να είναι επίσης πάνω από 990 λεπτά). Επιλέξαμε να το κάνουμε αυτό, διότι η ιστοσελίδα δεν παρείχε πληροφορίες για τις καθυστερήσεις του αγώνα ενώ κάποιες αλλαγές ή κάποιες κάρτες δίνονται μετά το 90' λεπτό. Προκειμένου να το κάνουμε αυτό, διαβάζουμε τα αρχεία excel που δημιουργούνται μέσω αυτών των αρχείων της python, μέσω της βιβλιοθήκης panda όπως φαίνεται παρακάτω:

```
from bs4 import BeautifulSoup
import requests
import random as rd
import xlsxwriter as xw
import pandas as pd

#Here we write a code order to fix the participation time of every player in every team in a match, so that the team has a total participation time in match equal to 990 minutes

#We import the Player_statistics excel file that we created
players_stats = pd.read_excel('participates.xlsx')
df1=pd.DataFrame(players_stats).sort_values(by='match_ID',ascending=False,ignore_index=True)
#We import the substitutions excel file that we created in the subs.py file
subs = pd.read_excel('substitutes.xlsx',header=0)
df2=pd.DataFrame(subs)

#We import the attributesViolation excel file that we created in the card.py file
cards = pd.read_excel('attributes_violation.xlsx',header=0)
df3=pd.DataFrame(cards)
```

Εικόνα 8: Διόρθωση πινάκων-μέρος Α

και κάνουμε τα εξής βήματα:

- Για κάθε μία από τις αλλαγές, αλλάζουμε τον χρόνο της αλλαγής με βάση το συμμετοχής του παίκτη στον αγώνα. Πιο συγκεκριμένα, επειδή ο τρόπος αποθήκευσης των δεδομένων των αλλαγών στο αρχείο excel με όνομα substitutes ήταν τέτοιος ώστε ανά δύο να περιγράφουν μία αλλαγή παίκτη με έναν άλλον (ο πρώτος σε κάθε αλλαγή είναι αυτός που βγαίνει, ο δεύτερος είναι αυτός που μπαίνει) , φροντίσαμε το άθροισμα των δύο χρόνων συμμετοχής των παικτών που συμμετέχουν στην αλλαγή να είναι ίσος με 90 λεπτά. Με βάση αυτό, τροποποιούμε τα λεπτά έναρξης και λήξης συμμετοχής τους στον αγώνα. Από αυτούς, διαγράφουμε ορισμένους παίκτες από κάποιους αγώνες, λόγω του ότι η ιστοσελίδα είχε ορισμένες εκτελέσεις πέναλτι, αποκρούσεις πέναλτι και αυτογκόλ ως αλλαγές.

```
p=[]
participation_ID=0
#To fix the participation time of the substitutes
for j in range(0,len(df2)):# For every player in the substitutions file
    for i in range(0,len(df1)):#For every player in the Player_statistics file
        if j%2==0:#we have the start of participation
            if int(df1.at[i,'player_ID'])==int(df2.at[j,'player_ID']):
                if int(df1.at[i,'minute_in_match'])-1==end#of participation time is when the substitution happens
                    end+=90:#if the substitution happens after the 90 minutes we make it happen in the 89' minute
                    end-=9
                else:#initialization of start of participation as 0 minutes
                    match_ID=df1.at[i,'match_ID']+match_ID
                    player_ID=int(df2.at[j,'player_ID'])+player_ID
                    participation=int(df1.at[i,'participation_in_match'])#participation in match for that player
                    #Here we fix the participation time and start and end time of the players
                    if participation>90:#if that condition is true,
                        if participation>[participation_ID-1][6]:# then if the participation of the Player that left is more than that of the Player that got in, then that means that he was a participant
                            participation=[participation_ID-1][6]-90
                            start=end-participation
                        else:#else if the participation of the Player that left is less than that of the Player that got in, then we need to fix the participation time of the player that got in
                            p[participation_ID-1][6]-=participation+p[participation_ID-1][6]-90
                            p[participation_ID-1][4]=p[participation_ID-1][5]-p[participation_ID-1][6]
                    end=participation#participation time determines finally the end time of the player that left the game
                    try:
                        p.append([participation_ID,match_ID,player_ID,start,end,participation,df1.at[i,'goals'],df1.at[i,'saves'],\
                                int(df1.at[i,'shots']),int(df1.at[i,'shots_on_target']),int(df1.at[i,'assists']),int(df1.at[i,'tackles']),int(df1.at[i,'passes']),df1.at[i,'position']])
                    except:
                        p.append([participation_ID,match_ID,player_ID,start,end,participation,df1.at[i,'start'],int(df1.at[i,'goals']),'NULL',int(df1.at[i,'shots']),\
                                int(df1.at[i,'shots_on_target']),int(df1.at[i,'assists']),int(df1.at[i,'tackles']),int(df1.at[i,'passes']),df1.at[i,'position']])
                    participation_ID+=1
                    continue
            else:#for the players that got into the game
                if(int(df1.at[i,'player_ID'])==int(df2.at[j,'player_ID'])) and int(df1.at[i,'match_ID'])==int(df2.at[j,'match_ID']):
                    end+=int(df2.at[j,'minute_in_match'])+int(df1.at[i,'participation_in_match'])-1#end of participation time is when the substitution happens plus the participation of the player
                    if end>90:#if his end time is over 90 make it 90
                        end=90
                    match_ID=int(df1.at[i,'match_ID'])+match_ID
                    player_ID=int(df2.at[j,'player_ID'])+player_ID
                    participation=int(df1.at[i,'participation_in_match'])#participation in match for the Player
                    start=end-participation#initialization of start time as end time-participation
                    starter=0#he is not a starter
                    starter+=1
                    try:
                        p.append([participation_ID,match_ID,player_ID,start,end,participation,df1.at[i,'start'],int(df1.at[i,'goals']),df1.at[i,'saves'],\
                                int(df1.at[i,'shots']),int(df1.at[i,'shots_on_target']),int(df1.at[i,'assists']),int(df1.at[i,'tackles']),int(df1.at[i,'passes']),df1.at[i,'position']])
                    except:
                        p.append([participation_ID,match_ID,player_ID,start,end,participation,df1.at[i,'start'],int(df1.at[i,'goals']),'NULL',int(df1.at[i,'shots']),\
                                int(df1.at[i,'shots_on_target']),int(df1.at[i,'assists']),int(df1.at[i,'tackles']),int(df1.at[i,'passes']),df1.at[i,'position']])
                    participation_ID+=1
                    continue
```

```
#To delete the players who have multiple ids
L=[]
M=[[22,263],[79,328],[84,221],[86,533],[89,493],[95,394],[126,639],[225,144],[235,533],[237,396],[238,118],[259,630],[277,263],[290,600],[307,632],[349,226]]
flags={}
flag1=0
for i in range(len(M)):
    flags.append(0)
for x in range(len(p)):
    for y in range(len(M)):# If a player gets/leaves a match with more than one ids (p[x][2]) in the match p[x][0] dont include the other id
        if p[x][1]==M[y][0] and p[x][2]==M[y][1]:
            flags[y]=1
            flag1=1
            if flags[y]==2:
                continue
            elif flags[y]==1:
                L.append(p[x])
    if flag1==0:
        L.append(p[x])
    else:
        flag1=0
```

Εικόνα 9: Διόρθωση πινάκων-μέρος Β

- 2) Για κάθε μία από τις κόκκινες και δεύτερες κίτρινες κάρτες που δόθηκαν, αλλάζουμε τον χρόνο που δόθηκαν ώστε να μην είναι μετά το 90° λεπτό και το κάνουμε 89.

```
#To fix the participation time of the attributesViolation relationship when a player gets a Second Yellow or a Red card
for j in range(0,len(df3)):
    for i in range(0,len(df1)):
        if("Y" in df1.at[i,'start']) and int(df1.at[i,'player_ID'])==int(df3.at[j,'player_ID']) and int(df1.at[i,'match_ID'])==int(df3.at[j,'match_ID']) and (df3.at[j,'card']=="Red" or \
        df3.at[j,'card']=="Second Yellow") and df1.at[i,'player_ID']!=596 or df1.at[i,'player_ID']!=538 or df1.at[i,'player_ID']!=172 or df1.at[i,'player_ID']!=370 \
        or df1.at[i,'player_ID']!=495 or df1.at[i,'player_ID']!=601 or df1.at[i,'player_ID']!=331 or df1.at[i,'player_ID']!=409 or df1):
            start=0#initialization of start of participation as 0 minutes
            end=(df3.at[j,'minute_in_match'])-1#initialization of end of participation as minute_in_match of attributes violation
            if end>90:#if the attribution happens after the 90 minutes we make it happen in the 89' minute
                end=89
            match_ID=int(df1.at[i,'match_ID'])
            player_ID=int(df3.at[j,'player_ID'])
            participationID+=1
            starter=df1.at[i,'start']
            try:
                L.append([participation_ID,match_ID,player_ID,starter,start,end,participation,df1.at[i,'start'],int(df1.at[i,'goals']),df1.at[i,'saves'],\
                    int(df1.at[i,'shots']),int(df1.at[i,'shots_on_target']),int(df1.at[i,'assists']),int(df1.at[i,'tackles']),int(df1.at[i,'passes']),df1.at[i,'position']])
            except:
                L.append([participation_ID,match_ID,player_ID,starter,start,end,participation,df1.at[i,'start'],int(df1.at[i,'goals']),'NULL',int(df1.at[i,'shots']),\
                    int(df1.at[i,'shots_on_target']),int(df1.at[i,'assists']),int(df1.at[i,'tackles']),int(df1.at[i,'passes']),df1.at[i,'position']])
            participation_ID+=1
        participation_ID+=1
```

Εικόνα 10: Διόρθωση πινάκων-μέρος Γ

- 3) Για κάθε έναν παίκτη που έπαιξε και τα 90 λεπτά, δεν αλλάζουμε κάτι

```
#To get the participation time of the players that participate the whole 90 minutes
for i in range(0,len(df1)):
    if(int(df1.at[i,'participation_in_match'])==90 and (df1.at[i,'player_ID'])==596 or df1.at[i,'player_ID']!=538 or df1.at[i,'player_ID']!=172 or df1.at[i,'player_ID']!=370 \
    or df1.at[i,'player_ID']!=495 or df1.at[i,'player_ID']!=601 or df1.at[i,'player_ID']!=331 or df1.at[i,'player_ID']!=409 \
    or df1.at[i,'player_ID']!=463 or df1.at[i,'player_ID']!=559 or df1.at[i,'player_ID']!=458):
        match_ID=int(df1.at[i,'match_ID'])
        player_ID=int(df1.at[i,'player_ID'])
        start=0#start_time
        end=90#end_time
        starter=1
        participation=90
        try:
            L.append([participation_ID,match_ID,player_ID,starter,start,end,participation,df1.at[i,'start'],int(df1.at[i,'goals']),df1.at[i,'saves'],\
                int(df1.at[i,'shots']),int(df1.at[i,'shots_on_target']),int(df1.at[i,'assists']),int(df1.at[i,'tackles']),int(df1.at[i,'passes']),df1.at[i,'position']])
        except:
            L.append([participation_ID,match_ID,player_ID,starter,start,end,participation,df1.at[i,'start'],int(df1.at[i,'goals']),'NULL',int(df1.at[i,'shots']),\
                int(df1.at[i,'shots_on_target']),int(df1.at[i,'assists']),int(df1.at[i,'tackles']),int(df1.at[i,'passes']),df1.at[i,'position']])
        participation_ID+=1
    participation_ID+=1
```

Εικόνα 11: Διόρθωση πινάκων-μέρος Δ

Από αυτά επίσης κρατάμε το ID του αγώνα και του παίκτη, το αν είναι βασικός ή όχι (μεταβλητή starter), τα φτιαγμένα λεπτά έναρξης και λήξης της συμμετοχής στον αγώνα καθώς και τον χρόνο συμμετοχής του στον αγώνα (start, end, participation), καθώς και τα διάφορα στατιστικά του παίκτη στον αγώνα, τα οποία και ορίστηκαν στο αρχείο participates, και τα αποθηκεύουμε όλα αυτά σε μία λίστα L.

Στην συνέχεια, δημιουργούμε καινούργια αρχεία για το πίνακα participates, attributesViolation και substitutes τα οποία και τελικά θα μπουν στη βάση μας.

Για τη δημιουργία αυτών, κάνουμε μία αντιστοίχιση των στοιχείων των παλιών αρχείων με αυτά της λίστας L, μέσω των player_IDs και match_IDs. Αναλυτικά:

- 1) Για κάθε ένα στοιχείο της λίστας L, ελέγχουμε αν υπάρχουν τα match και player IDs στο αρχείο participates, και αν υπάρχουν τότε τα στοιχεία της λίστας L μπαίνουν στο νέο αρχείο του participates.

```

headers = ['player_ID','match_ID','start','goals','participation_in_match','saves','shots','shots_on_target','assists','tackles','passes','position','\n'
           'start_of_participation','end_of_participation']
workbook = xw.Workbook('participates_fixed.xlsx')
worksheet = workbook.add_worksheet()
row=0
col=0
for count,l in enumerate(headers):
    worksheet.write(row, count,l)

for i in range(len(df1)):
    for j in range(len(L)):
        if(int(df1.at[i,'player_ID'])==L[j][2] and int(df1.at[i,'match_ID'])==L[j][1]):
            row+=1
            #stats
            print(row)
            worksheet.write(row,col,L[j][2])#player ID
            worksheet.write(row,col+1,L[j][1])#match ID
            worksheet.write(row,col+2,L[j][7]) #starter
            worksheet.write(row,col+3,L[j][8]) #goals
            worksheet.write(row,col+4,L[j][6]) #participation in match
            try:
                worksheet.write(row,col+5,df1.at[i,'saves'])
            except:
                worksheet.write(row,col+5,"NULL")
            worksheet.write(row,col+6,L[j][10]) #shots
            worksheet.write(row,col+7,L[j][11]) #shots on target
            worksheet.write(row,col+8,L[j][12]) #assists
            worksheet.write(row,col+9,L[j][13]) #tackles
            worksheet.write(row,col+10,L[j][14]) #passes
            worksheet.write(row,col+11,L[j][15]) #position
            worksheet.write(row,col+12,L[j][4]) #start of participation
            worksheet.write(row,col+13,L[j][5]) #end of participation
#We finish creating the excel file.
workbook.close()

```

Εικόνα 12: Διόρθωση πινάκων-μέρος Ε

- 2) Για κάθε ένα στοιχείο της λίστας L, ελέγχουμε αν υπάρχουν τα match και player IDs στο αρχείο substitutes και αν ο παίκτης μπαίνει ή βγαίνει από το αγώνα. Αν ο παίκτης που συμμετέχει στην αλλαγή, μπαίνει μέσα στον αγωνιστικό χώρο, τότε ο χρόνος της αλλαγής, είναι αυτός της έναρξης της συμμετοχής του, ενώ αν βγαίνει είναι ο χρόνος λήξης της συμμετοχής. Τέλος, αποθηκεύουμε στο νέο αρχείο του substitutes πίνακα, το substitution_ID του, το match_ID, το ID του παίκτη καθώς και το λεπτό της αλλαγής.

```

headers = ['substitution_ID','match_ID','player_ID','minute_in_match']
workbook = xw.Workbook('substitutions_fixed.xlsx')
worksheet = workbook.add_worksheet()
row=0
col=0
for count,l in enumerate(headers):
    worksheet.write(row, count,l)

#We fix the minute_in_match in substitutes
for x in range(len(df2['player_ID'])):
    for y in range(len(L)):
        #If a player gets into a game, then his start time is the time when the substitution happens
        if (df2['player_ID'][x]==L[y][2] and df2['match_ID'][x]==L[y][1] and x%2==0):
            minute_in_match=L[y][4]
            match_ID=df2['match_ID'][x]
            substitution_ID=df2['substitution_ID'][x]
            player_ID=df2['player_ID'][x]
        #If a player leaves a game, then his end time is the time when the substitution happens
        elif (df2['player_ID'][x]==L[y][2] and df2['match_ID'][x]==L[y][1] and x%2==1):
            minute_in_match=L[y][5]
            match_ID=df2['match_ID'][x]
            substitution_ID=df2['substitution_ID'][x]
            player_ID=df2['player_ID'][x]
        row+=1
        worksheet.write(row,col,substitution_ID)
        worksheet.write(row,col+1,match_ID)
        worksheet.write(row,col+2,player_ID)
        worksheet.write(row,col+3,minute_in_match)

#We finish creating the excel file.
workbook.close()

```

Εικόνα 13: Διόρθωση πινάκων-μέρος ΣΤ

3) Για κάθε ένα στοιχείο της λίστας L, ελέγχουμε αν υπάρχουν τα match και player IDs στο αρχείο attributesViolation και τι κάρτα πήρε ο παίκτης. Αν ο παίκτης πήρε κάρτα, τότε ελέγχουμε αν το λεπτό που πήρε τη κάρτα είναι μετά το 90^o λεπτό, οπότε και το κάνουμε 89, και στη συνέχεια βάζουμε υπόλοιπα στοιχεία που χρειάζεται ο πίνακας attributesViolation στο νέο αρχείο (violation_ID, match_ID, player_ID, disqualification, minute_in_match, card).

Αν η Κάρτα είναι δεύτερη κίτρινη ή Κόκκινη, τότε θέτουμε ως νέο λεπτό της καταγραφής της κάρτας, το λεπτό λήξης συμμετοχής του παίκτη και στη συνέχεια βάζουμε υπόλοιπα στοιχεία που χρειάζεται ο πίνακας attributesViolation στο νέο αρχείο (violation_ID, match_ID, player_ID, disqualification, minute_in_match, card).

Τέλος, αν δεν ισχύει τίποτα από τα παραπάνω, τότε απλά βάζουμε τα απαραίτητα στοιχεία της λίστας L στο νέο αρχείο των attributesViolation.

```
#We fix the minute_in_match in attributesViolation
for x in range(len(df3['minute_in_match'])):
    for y in range(len(L)): #if the player gets a Red or a Second Yellow card,then set the minute in match as the end time of the Player
        if (df3['player_ID'][x]==L[y][2] and df3['match_ID'][x]==L[y][1] and (df3['card'][x]=="Red" or df3['card'][x]=="Second Yellow")):
            minute_in_match=L[y][5]
            row+=1
            worksheet.write(row,col,df3.at[x,'violation_ID'])
            worksheet.write(row,col+1,df3.at[x,'match_ID'])
            worksheet.write(row,col+2,df3.at[x,'player_ID'])
            worksheet.write(row,col+3,df3.at[x,'disqualification'])
            worksheet.write(row,col+4,minute_in_match)
            worksheet.write(row,col+5,df3.at[x,'card'])

        elif (df3['player_ID'][x]==L[y][2] and df3['match_ID'][x]==L[y][1] and df3['minute_in_match'][x]>=90):#If the minute in which the violation happens is over 90', then make 89
            minute_in_match=89
            row+=1
            worksheet.write(row,col,df3.at[x,'violation_ID'])
            worksheet.write(row,col+1,df3.at[x,'match_ID'])
            worksheet.write(row,col+2,df3.at[x,'player_ID'])
            worksheet.write(row,col+3,df3.at[x,'disqualification'])
            worksheet.write(row,col+4,minute_in_match)
            worksheet.write(row,col+5,df3.at[x,'card'])

    #else
        if(df3['player_ID'][x]==L[y][2] and df3['match_ID'][x]==L[y][1] and df3['minute_in_match'][x]<90 and (df3['card'][x]!="Red" and df3['card'][x]!="Second Yellow")):
            row+=1
            worksheet.write(row,col,df3.at[x,'violation_ID'])
            worksheet.write(row,col+1,df3.at[x,'match_ID'])
            worksheet.write(row,col+2,df3.at[x,'player_ID'])
            worksheet.write(row,col+3,df3.at[x,'disqualification'])
            worksheet.write(row,col+4,df3.at[x,'minute_in_match'])
            worksheet.write(row,col+5,df3.at[x,'card'])

#Creation of new attributes violation relation file
workbook.close()
```

Εικόνα 14: Διόρθωση πινάκων-μέρος Z

Επίσης, λόγω του ότι εμφανίζονται παίκτες με διπλά IDs (λόγω μεταγραφής τους σε άλλες ομάδες) κρατήσαμε ένα από τα δύο IDs και διαγράψαμε τα άλλα ενώ κάναμε και άλλες μικρές διορθώσεις στα καινούργια αρχεία (τα οποία προέκυψαν λόγω σφαλμάτων στον hmtl κώδικα της ιστοσελίδας). Οι διορθώσεις αυτές φαίνονται αναλυτικά στις οδηγίες εγκατάστασης.

Αρχικοποίηση της βάσης δεδομένων και εισαγωγή των δεδομένων:

```
initialize_database_without.py - C:\Users\konst\Desktop\7o εξαμηνό\project\initialize_database_without.py (3.10.1)
File Edit Format Run Options Window Help
import sqlite3
import time
import csv
import pandas as pd

class DataModel(): #class that connects to the database and creates a cursor
    def __init__(self, filename):
        self.filename = filename
        try:
            self.con = sqlite3.connect(filename)
            self.con.row_factory = sqlite3.Row
            self.cursor = self.con.cursor()
            print("Successful connection to database", filename)
        except sqlite3.Error as error:
            print("Error connecting to the database sqlite", error)

    def close(self): #stop the connection to the database
        self.con.commit()
        self.con.close()

    def executeSQL(self, query, show=False): #execute SQL queries
        try:
            t1 = time.perf_counter()
            for statement in query.split(";"):
                if statement.strip():
                    self.cursor.execute(statement)
                    sql_time = time.perf_counter() - t1
                    print(f'The code was executed {statement[:40]}... in {sql_time:.5f} sec')
            if show:
                d=[]
                for row in self.cursor.fetchall():
                    d.append([str(item) for item in row])
            self.con.commit()
            return d
        except sqlite3.Error as error:
            print(f"Error while trying to execute SQL", error)
            return False

if __name__ == "__main__":
    dbfile = "project.db" #name of the database file that will be created
    d = DataModel(dbfile) #connecting to the database

    #create each table of the database:
    sql="""CREATE TABLE "Coach" (
        "coach_ID"      INTEGER,
        "first_name"    TEXT,
        "last_name"     TEXT NOT NULL,
        "contract_start_date" TEXT CHECK(contract_start_date IS strftime('%Y-%m-%d',contract_start_date)),
        "contract_end_date" TEXT CHECK(contract_end_date IS strftime('%Y-%m-%d',contract_end_date)),
        "team_code"     INTEGER,
        FOREIGN KEY("team_code") REFERENCES "Team"("team_ID") ON UPDATE CASCADE ON DELETE SET NULL,
        PRIMARY KEY("coach_ID" AUTOINCREMENT)
    );""" #creating the table Coach
    d.executeSQL(sql, show=True)

    sql="""CREATE TABLE "Match_"
        ("match_ID"      INTEGER,
        "score"         TEXT,
        "datetime_"     TEXT CHECK(datetime_ IS strftime('%Y-%m-%d %H:%M',datetime_)),
        "matchweek"     INTEGER CHECK(matchweek>=1 AND matchweek<=38),
        "stadium_code"  INTEGER,
        PRIMARY KEY("match_ID" AUTOINCREMENT),
        FOREIGN KEY("stadium_code") REFERENCES "Stadium"(stadium_ID) ON UPDATE CASCADE ON DELETE SET NULL
    );""" #creating the table Match_
    d.executeSQL(sql, show=True)

    sql="""CREATE TABLE "Player" (
        "player_ID"     INTEGER,
        "shirt_number"  INTEGER CHECK(shirt_number>0),
        "height"        INTEGER CHECK(height>0),
        "date_of_birth" TEXT CHECK(date_of_birth IS strftime('%Y-%m-%d',date_of_birth)),
        "first_name"    TEXT,
        "last_name"     TEXT NOT NULL,
        PRIMARY KEY("player_ID" AUTOINCREMENT)
    );""" #creating the table Player
    d.executeSQL(sql, show=True)

    sql="""CREATE TABLE "Referee" (
        "referee_ID"    INTEGER,
        "last_name"     TEXT NOT NULL,
        "first_name"    TEXT,
        "yellow_cards"  INTEGER CHECK(yellow_cards>=0),
        "red_cards"     INTEGER CHECK(red_cards>=0),
        "number_of_matches" INTEGER CHECK(number_of_matches>=0),
        PRIMARY KEY("referee_ID" AUTOINCREMENT)
    );""" #creating the table Referee
    d.executeSQL(sql, show=True)
```

```

sql='''CREATE TABLE "Stadium" (
    "stadium_ID"      INTEGER,
    "name"      TEXT NOT NULL,
    "date_built"    INTEGER,
    "place"      TEXT,
    "capacitance"   INTEGER CHECK(capacitance>0),
    PRIMARY KEY("stadium_ID" AUTOINCREMENT)
);'''#creating the table Stadium
d.executeSQL(sql, show=True)

sql='''CREATE TABLE "Team" (
    "team_ID"      INTEGER,
    "name"      TEXT NOT NULL,
    "founded"    TEXT,
    "wins"      INTEGER CHECK(wins>=0),
    "ties"      INTEGER CHECK(ties>=0),
    "losses"    INTEGER CHECK(losses>=0),
    "points"    INTEGER CHECK(points>=0),
    "stadium_ID"  INTEGER,
    FOREIGN KEY("stadium_ID") REFERENCES "Stadium"("stadium_ID") ON UPDATE CASCADE ON DELETE SET NULL,
    PRIMARY KEY("team_ID" AUTOINCREMENT)
);'''#creating the table Team
d.executeSQL(sql, show=True)

sql='''CREATE TABLE "attributesViolation" (
    "violation_ID"  INTEGER,
    "match_ID"      INTEGER,
    "player_ID"    INTEGER,
    "disqualification"  INTEGER CHECK(disqualification IN (0,1)),
    "minute_in_match"  INTEGER CHECK(minute_in_match>=0 AND minute_in_match<=90),
    "card"      TEXT CHECK(card IN ('Yellow','Second Yellow','Red')),
    FOREIGN KEY("match_ID") REFERENCES "Match"("match_ID") ON UPDATE CASCADE ON DELETE SET NULL,
    FOREIGN KEY("player_ID") REFERENCES "Player"("player_ID") ON UPDATE CASCADE ON DELETE SET NULL,
    PRIMARY KEY("match_ID","player_ID","violation_ID")
);'''#creating the table attributesViolation
d.executeSQL(sql, show=True)

sql='''CREATE TABLE "belongs" (
    "player_ID"    INTEGER,
    "team_ID"      INTEGER,
    "contract_start_day"  TEXT CHECK(contract_start_day IS strftime('%Y-%m-%d',contract_start_day)),
    "contract_end_day"  TEXT CHECK(contract_end_day IS strftime('%Y-%m-%d',contract_end_day)),
    "salary"      INTEGER CHECK(salary>0),
    FOREIGN KEY("team_ID") REFERENCES "Team"("team_ID") ON UPDATE CASCADE ON DELETE SET NULL,
    FOREIGN KEY("player_ID") REFERENCES "Player"("player_ID") ON UPDATE CASCADE ON DELETE SET NULL,
    PRIMARY KEY("player_ID","team_ID")
);'''#creating the table belongs
d.executeSQL(sql, show=True)

sql='''CREATE TABLE "enters" (
    "match_ID"    INTEGER,
    "team_ID"      INTEGER,
    "home"      INTEGER CHECK(home IN (0,1)),
    FOREIGN KEY("team_ID") REFERENCES "Team"("team_ID") ON UPDATE CASCADE ON DELETE SET NULL,
    FOREIGN KEY("match_ID") REFERENCES "Match"("match_ID") ON UPDATE CASCADE ON DELETE SET NULL,
    PRIMARY KEY("match_ID","team_ID")
);'''#creating the table enters
d.executeSQL(sql, show=True)

sql='''CREATE TABLE "inspects" (
    "match_ID"    INTEGER,
    "referee_ID"  INTEGER,
    "status"      TEXT CHECK(status IN ('Referee','AR1','AR2','4th')),
    FOREIGN KEY("match_ID") REFERENCES "Match"("match_ID") ON UPDATE CASCADE ON DELETE SET NULL,
    FOREIGN KEY("referee_ID") REFERENCES "Referee"("referee_ID") ON UPDATE CASCADE ON DELETE SET NULL,
    PRIMARY KEY("match_ID","referee_ID")
);'''#creating the table inspects
d.executeSQL(sql, show=True)

sql='''CREATE TABLE "participates" (
    "player_ID"    INTEGER,
    "match_ID"      INTEGER,
    "start"      TEXT CHECK(start IN ('Y','Y*','N*')),
    "goals"      INTEGER CHECK(goals>=0),
    "participation_in_match"  INTEGER CHECK(participation_in_match>0 AND participation_in_match<91),
    "saves"      INTEGER CHECK(saves>=0),
    "shots"      INTEGER CHECK(shots>=0),
    "shots_on_target"  INTEGER CHECK(shots_on_target>=0),
    "assists"    INTEGER CHECK(assists>=0),
    "tackles"    INTEGER CHECK(tackles>=0),
    "passes"      INTEGER CHECK(passes>=0),
    "position"    TEXT,
    "start_of_participation"  INTEGER CHECK(start_of_participation>=0 AND start_of_participation<90),
    "end_of_participation"  INTEGER CHECK(end_of_participation>0 AND end_of_participation<=90),
    PRIMARY KEY("match_ID","player_ID","participation_ID"),
    FOREIGN KEY("match_ID") REFERENCES "Match"("match_ID") ON UPDATE CASCADE ON DELETE SET NULL,
    FOREIGN KEY("player_ID") REFERENCES "Player"("player_ID") ON UPDATE CASCADE ON DELETE SET NULL
);'''#creating the table participates
d.executeSQL(sql, show=True)

```

```

sql='''CREATE TABLE "substitutes" (
    "substitution_ID"           INTEGER,
    "match_ID"                  INTEGER,
    "player_ID"                 INTEGER,
    "minute_in_match"           INTEGER CHECK(minute_in_match>0 AND minute_in_match<90),
    FOREIGN KEY("player_ID") REFERENCES "Player"("player_ID") ON UPDATE CASCADE ON DELETE SET NULL,
    FOREIGN KEY("match_ID") REFERENCES "Match"("match_ID") ON UPDATE CASCADE ON DELETE SET NULL,
    PRIMARY KEY("substitution_ID","player_ID","match_ID")
);'''#creating the table substitutes
d.executeSQL(sql, show=True)

#inserting excel and csv files to the database
players = pd.read_excel(r'excel_csv\Players.xlsx', sheet_name='Sheet1',header=0) #Player
players.to_sql('Player',d.con, if_exists='append', index=False)

coaches = pd.read_excel(r'excel_csv\Coach.xlsx', sheet_name='Sheet1',header=0) #Coach
coaches.to_sql('Coach',d.con, if_exists='append', index=False)

stadiums = pd.read_excel(r'excel_csv\Stadiums.xlsx', sheet_name='Sheet1',header=0) #Stadium
stadiums.to_sql('Stadium',d.con, if_exists='append', index=False)

matches = pd.read_excel(r'excel_csv\matches.xlsx', sheet_name='Sheet1',header=0) #Match_
matches.to_sql('Match_',d.con, if_exists='append', index=False)

referees = pd.read_excel(r'excel_csv\referee.xlsx', sheet_name='Sheet1',header=0) #Referee
referees.to_sql('Referee',d.con, if_exists='append', index=False)

teams = pd.read_csv(r'excel_csv\Team.csv', header=0) #Team
teams.to_sql('Team',d.con, if_exists='append', index=False)

violations = pd.read_excel(r'excel_csv\attributesViolation_fixed.xlsx', sheet_name='Sheet1',header=0) #attributes_violation
violations.to_sql('attributesViolation',d.con, if_exists='append', index=False)

belongs = pd.read_excel(r'excel_csv\Belongs.xlsx', sheet_name='Sheet1',header=0) #belongs
belongs.to_sql('belongs',d.con, if_exists='append', index=False)

enters = pd.read_excel(r'excel_csv\enters.xlsx', sheet_name='Sheet1',header=0) #enters
enters.to_sql('enters',d.con, if_exists='append', index=False)

inspects = pd.read_excel(r'excel_csv\inspects.xlsx', sheet_name='Sheet1',header=0) #inspects
inspects.to_sql('inspects',d.con, if_exists='append', index=False)

participates = pd.read_excel(r'excel_csv\participates_fixed.xlsx', sheet_name='Sheet1',header=0) #participates
participates.to_sql('participates',d.con, if_exists='append', index=False)

substitutes = pd.read_excel(r'excel_csv\substitutes.xlsx', sheet_name='Sheet1',header=0) #substitutes
substitutes.to_sql('substitutes',d.con, if_exists='append', index=False)

d.executeSQL('''create index position_index on participates(position)''', show=True)
d.executeSQL('''create index status_index on inspects(status)''', show=True)
d.executeSQL('''create index card_index on attributesViolation(card)''', show=True)
d.executeSQL('''create index matchweek_index on Match_(matchweek)''', show=True)
d.executeSQL('''create index f_lname_index on Player(first_name,last_name)''', show=True)

sql="""SELECT count(*),T.team_ID
      FROM Match_AS M,enters AS E,Team AS T
      WHERE M.match_ID=E.match_ID AND T.team_ID=E.team_ID AND ((CAST(SUBSTR(score, 1, 1) AS INTEGER)>CAST(SUBSTR(score, 3, 3) AS INTEGER) AND home=1)
      OR (CAST(SUBSTR(score, 1, 1) AS INTEGER)<CAST(SUBSTR(score, 3, 3) AS INTEGER) AND home=0))
      GROUP BY T.team_ID;""" #sql code for computing the number of wins of each team
wins=d.executeSQL(sql, show=True)

sql="""SELECT count(*),T.team_ID
      FROM Match_AS M,enters AS E,Team AS T
      WHERE M.match_ID=E.match_ID AND T.team_ID=E.team_ID AND CAST(SUBSTR(score, 1, 1) AS INTEGER)==CAST(SUBSTR(score, 3, 3) AS INTEGER)
      GROUP BY T.team_ID;""" #sql code for computing the number of ties of each team
ties=d.executeSQL(sql, show=True)

sql="""SELECT count(*),T.team_ID
      FROM Match_AS M,enters AS E,Team AS T
      WHERE M.match_ID=E.match_ID AND T.team_ID=E.team_ID AND ((CAST(SUBSTR(score, 1, 1) AS INTEGER)<CAST(SUBSTR(score, 3, 3) AS INTEGER) AND home=1)
      OR (CAST(SUBSTR(score, 1, 1) AS INTEGER)>CAST(SUBSTR(score, 3, 3) AS INTEGER) AND home=0))
      GROUP BY T.team_ID;""" #sql code for computing the number of losses of each team
losses=d.executeSQL(sql, show=True)

sql='''SELECT sum(IIF((CAST(SUBSTR(score, 1, 1) AS INTEGER))==(CAST(SUBSTR(score, 3, 3) AS INTEGER)),1,IIF(home,IIF((CAST(SUBSTR(score, 1, 1) AS INTEGER))
>(CAST(SUBSTR(score, 3, 3) AS INTEGER)),3,0),IIF((CAST(SUBSTR(score, 1, 1) AS INTEGER))>(CAST(SUBSTR(score, 3, 3) AS INTEGER)),0,3))) AS pointss,T.team_ID
      FROM Match_AS M,enters AS E,Team AS T
      WHERE M.match_ID=E.match_ID AND T.team_ID=E.team_ID
      GROUP BY T.team_ID
      ORDER BY pointss DESC;''' #sql code for computing the points of each team
points=d.executeSQL(sql, show=True)

#update wins/losses/ties of each team
for i in range(20):
    sql="""UPDATE Team
    SET wins={},ties={},losses={},points={}
    WHERE team_ID={};""".format(wins[i][0],ties[i][0],losses[i][0],points[i][0],i)
    d.executeSQL(sql, show=True)


```

```

sql="""SELECT referee_ID,count(*) as appearances
FROM Referee NATURAL JOIN inspects
GROUP BY referee_ID
ORDER BY referee_ID"""
appearances=d.executeSQL(sql, show=True)

sql="""SELECT referee_ID,count(*) as yellow_cards
FROM (Referee NATURAL JOIN inspects)NATURAL JOIN attributes_violation
WHERE card="Yellow" AND status="Referee"
GROUP BY referee_ID"""
yellow=d.executeSQL(sql, show=True)

sql="""SELECT referee_ID,count(*) as red_cards
FROM (Referee NATURAL JOIN inspects)NATURAL JOIN attributes_violation
WHERE (card="Red" OR card="Second Yellow") AND status="Referee"
GROUP BY referee_ID"""
red=d.executeSQL(sql, show=True)

for i in range(len(appearances)):
    sql="""UPDATE Referee
    SET number_of_matches={}
    WHERE referee_ID={};""".format(appearances[i][1],appearances[i][0])
    d.executeSQL(sql, show=True)

for i in range(len(yellow)):
    sql="""UPDATE Referee
    SET yellow_cards={}
    WHERE referee_ID={};""".format(yellow[i][1],yellow[i][0])
    d.executeSQL(sql, show=True)

for i in range(len(red)):
    sql="""UPDATE Referee
    SET red_cards={}
    WHERE referee_ID={};""".format(red[i][1],red[i][0])
    d.executeSQL(sql, show=True)

d.close()#closing the connection to the database

```

Εικόνα 15: Αρχικοποίηση βάσης και εισαγωγή δεδομένων

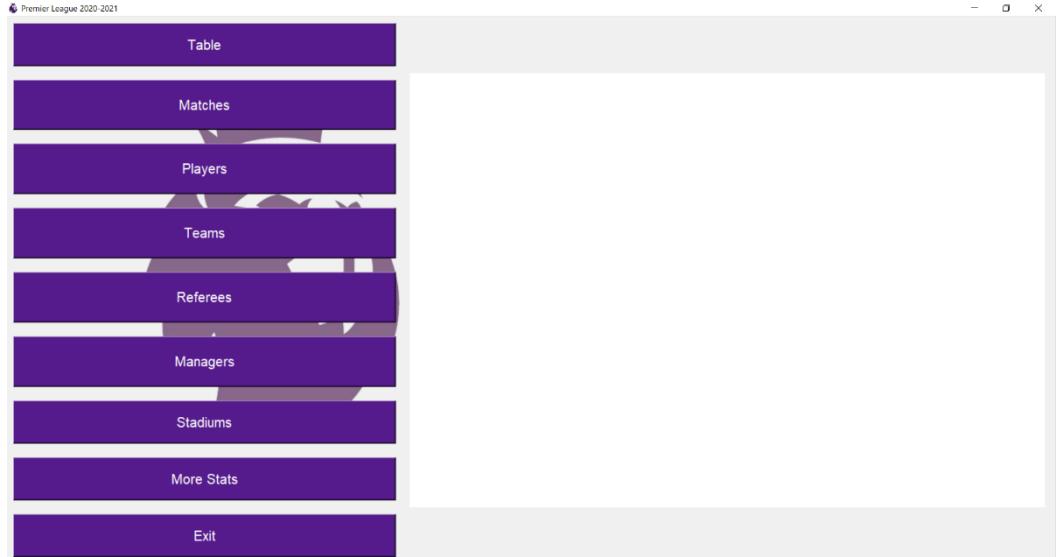
Σχολιασμός κώδικα:

Αρχικά φτιάχνουμε την κλάση DataModel, μέσω της οποίας κάνουμε τη σύνδεση και γενικότερα αλληλοεπιδράμε με τη βάση δεδομένων μας στην sqlite. Έπειτα θα δημιουργήσουμε το project μας με filename: project.db και σε αυτό δημιουργούμε τους πίνακες σύμφωνα με το σχεσιακό σχήμα, βάζοντας τους κατάλληλους περιορισμούς, όπως τα ξένα κλειδιά, να έχουν οι ημερομηνίες κατάλληλο format, κάποια από τα γνωρίσματα που παίρνουν συγκεκριμένες τιμές (π.χ. Boolean) να μην μπορούν να πάρουν κάποια άλλη τιμή και κάποιους περιορισμούς στα όρια τιμών των ακεραίων. Έπειτα μέσω της βιβλιοθήκης pandas φορτώνουμε τα δεδομένα από τα excel/csv αρχεία και τα περνάμε στη βάση μας. Στη συνέχεια για να βελτιώσουμε τους χρόνους σε κάποια queries δημιουργήσαμε indexes για τα γνωρίσματα κάποιων πινάκων οι οποίοι είχαν πολλά δεδομένα (αποφασίσαμε να βάλουμε indexes σε πίνακες με πολλά δεδομένα, στα γνωρίσματα τους που χρησιμοποιούνται συχνά σε queries της εφαρμογής μας). Τέλος υπολογίζουμε τα υπολογιζόμενα γνωρίσματα του πίνακα Team και του πίνακα Referee μέσω κάποιων queries.

Εφαρμογή:

Στη συνέχεια, έγινε η σύνταξη του κώδικα για τη δημιουργία της εφαρμογής του τοπικού πρωταθλήματος. Για την δημιουργία της εφαρμογής χρησιμοποιήσαμε τη βιβλιοθήκη tkinter της python, ώστε να έχουμε μία πιο εύκολη για τον χρήστη, επίδειξη των δεδομένων της βάσης μας.

Εκτελώντας τον κώδικα της εφαρμογής βλέπουμε την πρώτη σελίδα η οποία φαίνεται παρακάτω:



Εικόνα 16: Αρχική οθόνη εφαρμογής

Μέσω αυτής, ο χρήστης μπορεί πατώντας κάποιο από τα κουμπιά να δει διάφορα στοιχεία που τον ενδιαφέρουν σχετικά με το ποδοσφαιρικό πρωτάθλημα. Τα στοιχεία αυτά δημιουργούνται μέσω queries και τα αποθηκεύουμε σε λίστες της python ώστε να είναι πιο εύκολη η χρήση τους για τη δημιουργία της εφαρμογής μας. Παρακάτω φαίνεται αναλυτικά η λειτουργία του καθενός κουμπιού της εφαρμογής, καθώς και ορισμένες τυπικές αναζητήσεις και τροποποιήσεις που μπορούμε να κάνουμε στη βάση μας.

Πατώντας το κουμπί Table, εκτελείται η συνάρτηση buttonPushed1() και ο χρήστης μπορεί να δει στο διπλανό Frame το βαθμολογικό πίνακα του πρωταθλήματος. Ο χρήστης επιλέγει την αγωνιστική στην οποία θέλει να δει τον βαθμολογικό πίνακα και στη συνέχεια δημιουργείται ο βαθμολογικός πίνακας όπου μπορεί να δει την κατάταξη της κάθε ομάδας, τους βαθμούς της κάθε ομάδας καθώς και τις νίκες, τις ήττες και τις ισοπαλίες της ομάδας μέχρι αυτή την αγωνιστική. Το query που εκτελεί αυτή τη λειτουργία φαίνεται παρακάτω:

```
SELECT name, count(*) as number_of_matches,
       sum(IIF(CAST(SUBSTR(score,1,1) as integer)>CAST(SUBSTR(score,3,3) as integer) and e.home=1,1,0)) + sum(IIF(CAST(SUBSTR(score,1,1) as integer)<CAST(SUBSTR(score,3,3) as integer) and e.home=0,1,0)) as wins,
       sum(IIF(CAST(SUBSTR(score,1,1) as integer)=CAST(SUBSTR(score,3,3) as integer) and (e.home=1 or e.home=0),1,0)) as ties,
       sum(IIF(CAST(SUBSTR(score,1,1) as integer)<CAST(SUBSTR(score,3,3) as integer) and e.home=1,1,0)) + sum(IIF(CAST(SUBSTR(score,1,1) as integer)>CAST(SUBSTR(score,3,3) as integer) and e.home=0,1,0)) as losses,
       sum(IIF((CAST(SUBSTR(score, 1, 1) AS INTEGER))==(CAST(SUBSTR(score, 3, 3) AS INTEGER)),1,IIF(home,IIF((CAST(SUBSTR(score, 1, 1) AS INTEGER))>(CAST(SUBSTR(score, 3, 3) AS INTEGER)),3,0),IIF((CAST(SUBSTR(score, 1, 1) AS INTEGER))>(CAST(SUBSTR(score, 3, 3) AS INTEGER)),0,3)))) AS pointss
FROM Match_ AS M,enters AS E,Team AS T
WHERE M.match_ID=E.match_ID AND T.team_ID=E.team_ID AND matchweek<={ }
GROUP BY T.team_ID
ORDER BY pointss DESC,wins DESC,ties DESC ,losses DESC;
```

Όπου στην αγκύλη μπαίνει η επιλογή του χρήστη

Το αποτέλεσμα για την 38^η αγωνιστική φαίνεται παρακάτω:

The screenshot shows the Premier League 2020-2021 website's navigation menu on the left and the 'Table' section on the right. The table lists the top 20 teams with their positions, club names, played games, wins, draws, losses, and points. The table is titled 'Table of Premier League' and has a dropdown menu for 'matchweek 38'.

Position	Club	Played	Wins	Draws	Loses	Points
1	Manchester City	38	27	5	6	86
2	Manchester United	38	21	11	6	74
3	Liverpool	38	20	9	9	69
4	Chelsea	38	19	10	9	67
5	Leicester City	38	20	6	12	66
6	West Ham United	38	19	8	11	65
7	Tottenham Hotspur	38	18	8	12	62
8	Arsenal	38	18	7	13	61
9	Leeds United	38	18	5	15	59
10	Everton	38	17	8	13	59
11	Aston Villa	38	16	7	15	55
12	Newcastle United	38	12	9	17	45
13	Wolverhampton Wanderers	38	12	9	17	45
14	Crystal Palace	38	12	8	18	44
15	Southampton	38	12	7	19	43
16	Brighton & Hove Albion	38	9	14	15	41
17	Burnley	38	10	9	19	39
18	Fulham	38	5	13	20	28
19	West Bromwich Albion	38	5	11	22	26
20	Sheffield United	38	7	2	29	23

Εικόνα 17: Βαθμολογικός πίνακας

Πατώντας το κουμπί Matches, εκτελείται η συνάρτηση buttonPushed2() όπου ο χρήστης μπορεί να δει τους διάφορους αγώνες του πρωταθλήματος. Ο χρήστης επιλέγει την αγωνιστική στην οποία θέλει να δει τους αγώνες και στη συνέχεια δημιουργούνται οι αγώνες που γίνονται στην επιλεγμένη αγωνιστική.

The screenshot shows the Premier League 2020-2021 website's navigation menu on the left and the 'Matches' section on the right. The table lists the fixtures for matchweek 35, including the date/time, fixtures, score, stadium, and match report link. The table is titled 'Matches of Premier League' and has a dropdown menu for 'matchweek 35'.

datetime	Fixtures	Score	Stadium	Match Report
2021-05-07 20:00	Leicester City vs Newcastle United	2-4	King Power Stadium	Match Report
2021-05-09 12:30	Leeds United vs Tottenham Hotspur	3-1	Elland Road	Match Report
2021-05-09 15:00	Sheffield United vs Crystal Palace	0-2	Bramall Lane	Match Report
2021-05-09 17:30	Manchester City vs Chelsea	1-2	Etihad Stadium	Match Report
2021-05-09 20:15	Liverpool vs Southampton	2-0	Anfield	Match Report
2021-05-09 12:00	Wolverhampton Wanderers vs Brighton & Hove Albion	2-1	Molineux Stadium	Match Report
2021-05-09 14:05	Aston Villa vs Manchester United	1-3	Villa Park	Match Report
2021-05-09 19:30	West Ham United vs Everton	0-1	London Stadium	Match Report
2021-05-09 19:00	Arsenal vs West Bromwich Albion	3-1	Emirates Stadium	Match Report
2021-05-10 20:00	Fulham vs Burnley	0-2	Craven Cottage	Match Report

Εικόνα 18: Match αγωνιστικής

Ο χρήστης μπορεί να δει για κάθε αγώνα της συγκεκριμένης αγωνιστικής την ημερομηνία και την ώρα του αγώνα, τις ομάδες που συμμετέχουν στον αγώνα, το σκορ το αγώνα, το γήπεδο στο οποίο γίνεται ο αγώνας καθώς και ένα κουμπί Match Report στο οποίο ο χρήστης μπορεί να δει αναλυτικά τα στοιχεία του αγώνα και τα στατιστικά των παικτών στο συγκεκριμένο αγώνα.



Εικόνα 19: Match Report Αγώνα Μέρος Α

Στο Match Report, φαίνονται οι δύο ομάδες που έπαιξαν στον επιλεγμένο αγώνα, τα γκολ που έβαλε κάθε ομάδα οι προπονητές των δύο ομάδων στον συγκεκριμένο αγώνα, οι διαιτητές που διαιτήτευσαν τον αγώνα, οι παίκτες που σκόραραν γκολ στον αγώνα, οι παίκτες που πήραν κάρτα στον αγώνα αυτό καθώς και οι αλλαγές που έγιναν στον συγκεκριμένο αγώνα.

Για την εύρεση του αριθμού των γκολ που έβαλε η κάθε ομάδα σε έναν αγώνα, εκτελέσαμε τον ακόλουθο SQL κώδικα:

```

select a1.match_ID,a1.team1,a1.team_ID,a2.team2,a2.team_ID,a1.goals1,a1.goals,
IIF(a1.goals1==a1.goals,0,a1.goals1-a1.goals) as
autogoals1,a2.goals2,a2.goals,IIF(a2.goals2==a2.goals,0,a2.goals2-a2.goals) as
autogoals2,IIF(a1.goals1==a1.goals,0,a1.goals1-a1.goals) +a1.goals as
total_goals1,a2.goals+IIF(a2.goals2==a2.goals,0,a2.goals2-a2.goals) as total_goals2

from(select m.match_ID,IIF(e.home==1,t.name,NULL) as team1,t.team_ID,sum(goals) as
goals,cast(substr(score,1,1) as integer) as goals1

from Match_ as m,enters as e,Team as t,belongs as b,participates as p1

where m.match_ID=e.match_ID and t.team_ID=e.team_ID and b.team_ID=e.team_ID and
p1.match_ID=m.match_ID and p1.player_ID=b.player_ID and e.home=1 and
b.contract_end_day>m.datetime_ and m.match_ID={}

group by m.match_ID,t.team_ID

order by m.match_ID) as a1 left join

(select m.match_ID,IIF(e.home==0,t.name,NULL) as team2,t.team_ID,sum(goals) as
goals,cast(substr(score,3,3) as integer) as goals2

from Match_ as m,enters as e,Team as t,belongs as b,participates as p1

where m.match_ID=e.match_ID and t.team_ID=e.team_ID and b.team_ID=e.team_ID and
p1.match_ID=m.match_ID and p1.player_ID=b.player_ID and e.home=0 and
b.contract_end_day>m.datetime_ and m.match_ID={}

group by m.match_ID,t.team_ID

order by m.match_ID) as a2 on a1.match_ID=a2.match_ID

where a1.team1 is not NULL and a2.team2 is not NULL

order by a1.match_ID

```

Ο κώδικας αυτός, βγάζει σε μία γραμμή τις ομάδες που έπαιξαν σε ένα αγώνα, τα γκολ που έβαλαν με βάση το σκορ του αγώνα, καθώς και τα γκολ που έβαλαν με βάση τα στατιστικά των παικτών τους στον αγώνα. Ακόμη, επειδή σε ορισμένους αγώνες υπήρχαν και αυτογκολ, βρίκουμε και αυτά και σε περίπτωση που έχει μπει αυτογκολ, φαίνεται και αυτό στο πλαίσιο Scorers των Match Reports ως Own goals και αριθμός τους.

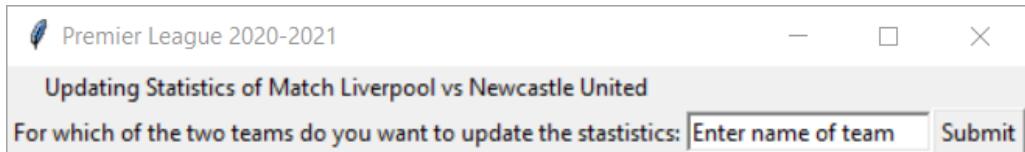
Manchester City-Stats											
Player	shirt_number	start	participation	position	goals	shots	shots on target	assists	saves	passes	tackles
Oleksandr Zinchenko	14	N	11	DF	0	0	0	0	0	9	0
İlkay Gündoğan	7	N	20	MF	0	2	0	0	0	14	1
Phil Foden	13	N	21	FW, MF	0	1	0	0	0	5	2
Sergio Agüero	20	Y	69	FW, MF	0	1	1	1	0	11	2
Ferán Torres	15	Y	70	FW, MF	0	1	0	0	0	18	0
Benjamin Mendy	18	Y	79	DF	0	1	0	0	0	29	0
Ederson	1	Y	90	GK	0	0	0	0	7	28	0
Rúben Dias	2	Y	90	DF	0	0	0	0	0	61	2
Rodri	3	Y	90	MF	0	2	0	0	0	68	3
Raheem Sterling	4	Y*	90	FW	1	3	2	0	0	35	0
João Cancelo	5	Y	90	DF	0	1	0	0	0	40	2
Gabriel Jesus	10	Y	90	FW	0	3	0	0	0	29	0
Aymeric Laporte	16	Y	90	DF	0	0	0	0	0	91	2
Nathan Aké	19	Y	90	DF	0	0	0	0	0	60	6
Total:14			990		1	15	3	1	7	498	20

Chelsea-Stats											
Player	shirt_number	start	participation	position	goals	shots	shots on target	assists	saves	passes	tackles
Callum Hudson-Odoi	19	N	15	FW, DF	0	0	0	0	0	4	0
Jorginho	100	N	23	MF	0	0	0	0	0	18	0
Andreas Christensen	15	Y	44	DF	0	0	0	0	0	22	0
Kurt Zouma	10	N	46	DF	0	0	0	0	0	36	1
N'Golo Kanté	7	Y	67	MF	0	0	0	0	0	39	2
Hakim Ziyech	16	Y	75	FW, MF	1	3	3	0	0	29	1
Edouard Mendy	2	Y	90	GK	0	0	0	0	5	37	0
Timo Werner	3	Y	90	FW	0	1	0	1	0	11	0
Reece James	5	Y	90	DF	0	3	0	0	0	44	2
César Azpilicueta	6	Y*	90	DF	0	0	0	1	0	74	5
Antonio Rüdiger	12	Y	90	DF	0	1	0	0	0	68	1
Christian Pulisic	13	Y	90	FW, MF	0	1	0	0	0	37	0
Marcos Alonso	18	Y	90	DF	1	2	2	0	0	35	1
Billy Gilmour	22	Y	90	MF	0	1	0	0	0	64	0
Total:14			990		2	12	5	2	5	518	13

Update

Εικόνα 20: Match Report Αγώνα Μέρος Β

Ο χρήστης ακόμη μπορεί να δει τα διάφορα στατιστικά των παικτών της κάθε ομάδας που συμμετέχει στον αγώνα. Εδώ δίνεται επίσης και η δυνατότητα τροποποίησης ορισμένων στατιστικών των παικτών. Πατώντας το κουμπί Update, εκτελείται η συνάρτηση Update όπου ο χρήστης βάζει την ομάδα στην οποία θέλει να τροποποιήσει τα στατιστικά των παικτών(πρέπει να είναι μία εκ των δύο που συμμετέχουν στον συγκεκριμένο αγώνα),



Εικόνα 21: Update Στατιστικών Αγώνα Μέρος Α

Και πατώντας το κουμπί Submit (έχοντας βάλει σωστό input) , εμφανίζεται το ακόλουθο παράθυρο :

Premier...

Player firstname

Player lastname

start

participation

position

shots

shots on target

assists

saves

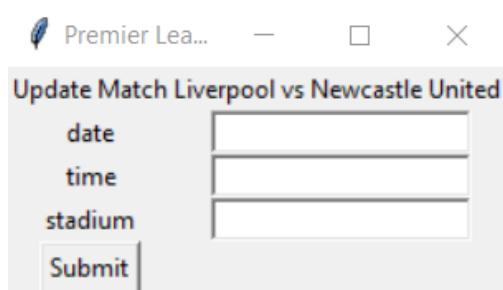
passes

tackles

Εικόνα 22: Update Στατιστικών Αγώνα Μέρος Β

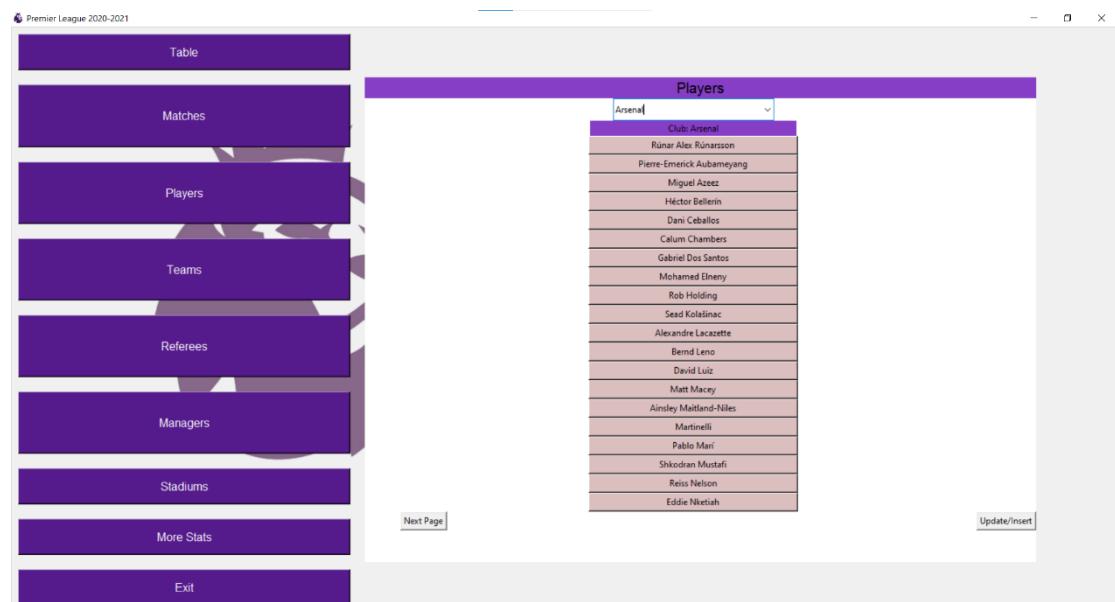
Στο οποίο ο χρήστης μπορεί να βάλει το όνομα που παίκτη για τον οποίο θέλει να τροποποιήσει τα στατιστικά του(πρέπει να συμμετέχει στο συγκεκριμένο αγώνα) και τα νέα στατιστικά του. Πατώντας το κουπί Update, εκτελείται η συνάρτηση UpdateButton() η οποία ελέγχει αν τα inputs είναι σωστά και αν είναι ανανεώνει τη βάση.

Τέλος, πατώντας το κουμπί Update Match , μπορούμε να τροποποιήσουμε τα δεδομένα των αγώνων (ημερομηνία, ώρα , γήπεδο στο οποίο γίνεται ο αγώνας).



Εικόνα 23: Update Δεδομένων Αγώνα

Πατώντας το κουμπί Players, εκτελείται η συνάρτηση buttonPushed3() όπου ο χρήστης μπορεί να δει τους διάφορους παίκτες του πρωταθλήματος. Διαλέγοντας μία από τις ομάδες του πρωταθλήματος, ο χρήστης μπορεί να δει τους διάφορους παίκτες της επιλεγμένης ομάδας.



Εικόνα 24: Παίκτες Ομάδας

Πατώντας σε κάποιον από αυτούς, ο χρήστης μπορεί να δει διάφορες πληροφορίες για αυτούς, όπως το ύψος τους, ημερομηνία γέννησης, έναρξη και λήξη συμβολαίου, καθώς και τα στατιστικά του παίκτη στους διάφορους αγώνες που έπαιξε

Pierre-Emerick Aubameyang													
Info	height: 187cm	date_of_birth: 1989-06-18	contract start date: 2019-01-24	possible contract end date: 2024-01-24	participation	goals	saves	shots	shots on target	assists	tackles	passes	
datetime	Opponent	Position	starter										
2020-09-12 12:30	Fulham	FW	Y*	90	1	None	3	2	0	0	0	21	
2020-09-19 20:00	West Ham United	FW	Y*	90	0	None	1	1	1	0	0	27	
2020-09-28 20:00	Liverpool	FW	Y*	90	0	None	0	0	0	0	2	11	
2020-10-04 14:00	Sheffield United	FW	Y*	90	0	None	2	2	0	0	0	26	
2020-10-17 17:30	Manchester City	FW	Y*	90	0	None	0	0	0	0	1	11	
2020-10-25 19:15	Leicester City	FW	Y*	90	0	None	2	0	0	0	2	20	
2020-11-01 16:30	Manchester United	FW	Y*	86	1	None	1	0	0	0	2	32	
2020-11-08 19:15	Aston Villa	FW	Y*	90	0	None	0	0	0	0	1	25	
2020-11-22 16:30	Leeds United	FW	Y*	90	0	None	3	1	0	0	0	16	
2020-11-29 19:15	Wolverhampton Wanderers	FW	Y*	90	0	None	5	1	0	0	0	9	
2020-12-06 16:30	Tottenham Hotspur	FW	Y*	90	0	None	2	0	0	0	0	13	
2020-12-13 19:15	Burnley	FW	Y*	90	0	None	4	2	0	0	2	25	
2020-12-16 18:00	Southampton	FW	Y*	90	1	None	1	1	0	0	0	18	
2020-12-29 18:00	Brighton & Hove Albion	FW	Y*	90	0	None	3	1	0	0	0	19	
2021-01-02 20:00	West Bromwich Albion	FW	Y*	90	0	None	3	1	0	0	0	35	
2021-01-14 20:00	Crystal Palace	FW	Y*	90	0	None	2	1	0	0	0	28	
2021-01-18 20:00	Newcastle United	FW	Y*	78	2	None	5	2	0	0	0	19	
2021-02-02 18:00	Wolverhampton Wanderers	FW	N	30	0	None	1	0	0	0	1	8	
2021-02-06 12:30	Aston Villa	FW	N	32	0	None	1	0	0	0	0	4	
2021-02-14 16:30	Leeds United	FW	Y*	90	3	None	4	2	0	0	0	18	
2021-02-21 16:30	Manchester City	FW	Y*	90	0	None	0	0	0	0	0	10	
2021-02-28 12:00	Leicester City	FW	N	7	0	None	1	0	0	0	0	2	
2021-03-06 12:30	Burnley	FW	Y*	90	1	None	4	1	0	0	0	17	
2021-03-21 15:00	West Ham United	FW	Y*	80	0	None	0	0	0	0	1	17	
2021-04-03 20:00	Liverpool	FW	Y*	76	0	None	0	0	0	0	4	20	
2021-05-02 14:00	Newcastle United	FW	Y*	77	1	None	4	1	1	0	0	23	
2021-05-12 20:15	Chelsea	FW	Y*	78	0	None	0	0	0	1	1	12	
2021-05-19 19:00	Crystal Palace	FW	Y*	90	0	None	2	0	0	0	0	15	
2021-05-23 16:00	Brighton & Hove Albion	FW	Y*	78	0	None	2	0	0	0	0	17	

Εικόνα 25: Πληροφορίες και Στατιστικά Παίκτη

Ο κώδικας SQL που κάνει αυτή τη διαδικασία είναι ο εξής:

Σε περίπτωση που παίκτης αυτός δεν έγινε μεταγραφή κατά τη διάρκεια της σεζόν :

```
select * from Player as p1,participates as p2,enters as e,Team as t,Match_ as m
where p1.player_ID=p2.player_ID and p2.match_ID=e.match_ID and e.team_ID=t.team_ID and
m.match_ID=e.match_ID and p1.player_ID={} and
t.team_ID!={}''.format(player_ID,team),
```

ενώ σε περίπτωση που έγινε μεταγραφή (και άρα ο ίδιος παίκτης υπάρχει σε δύο ομάδες) εκτελείται ο εξής κώδικας SQL:

```
select * from(select * from Player as p1,participates as p2,enters as e,Team as
t,Match_ as m
where p1.player_ID=p2.player_ID and p2.match_ID=e.match_ID and e.team_ID=t.team_ID
and m.match_ID=e.match_ID and p1.player_ID={}) as a1
left join (select * from Player as p1,participates as p2,enters as e,Team as t,Match_
as m
where p1.player_ID=p2.player_ID and p2.match_ID=e.match_ID and e.team_ID=t.team_ID
and m.match_ID=e.match_ID and p1.player_ID={}) as a2
on a1.match_ID=a2.match_ID
where a1.team_ID={} and a2.team_ID!=a1.team_ID.format(player_ID,player_ID,team)
```

όπου και γίνεται search ως προς τους αγώνες που έπαιξε για τη συγκεκριμένη ομάδα επιλογής.

Πατώντας το κουμπί Next Page, μπορούμε να δούμε και τους υπόλοιπους παίκτες της ομάδας, ενώ πατώντας το κουμπί Update/Insert μπορούμε να αλλάξουμε τα δεδομένα ορισμένων παικτών της επιλεγμένης ομάδας, καθώς και να προσθέσουμε παίκτες σε αυτές.

Premier L...

Update Players

Player firstname	
Player lastname	
shirt number	
height	
date of birth	
Contract start date:	
Contract end date:	
Salary:	

Submit

Εικόνα 26: Update/Insert Παίκτες σε Ομάδα

Αν ο παίκτης υπάρχει ήδη μέσα στην ομάδα επιλογής, τότε κάνει update τα δεδομένα του, ενώ αν δεν είναι τότε ενσωματώνεται στη βάση δεδομένων βάζοντας νέα στοιχεία στους πίνακες Player και belongs.

Πατώντας το κουμπί Teams, εκτελείται η συνάρτηση buttonPushed4() όπου ο χρήστης μπορεί να δει τις διάφορες ομάδες που συμμετέχουν στο πρωτάθλημα.



Εικόνα 27: Ομάδες Πρωταθλήματος

Πατώντας ένα από τα κουμπιά της Ομάδας, φαίνονται διάφορες πληροφορίες της επιλεγμένης ομάδας, όπως το όνομα, η ημερομηνία ίδρυσης, το συνολικό αποτέλεσμα της χρονιάς (νίκες-ισοπαλίες-ήττες) και την κατάταξη της στο πρωτάθλημα, καθώς και το γήπεδο το οποίο χρησιμοποιεί ως έδρα, ενώ υπάρχει ακόμη και δυνατότητα τροποποίησής τους.



Εικόνα 28: Πληροφορίες Ομάδας Πρωταθλήματος

Πατώντας το κουμπί Referees, εκτελείται η συνάρτηση buttonPushed5() όπου ο χρήστης μπορεί να δει τους διαιτητές του πρωταθλήματος, τον αριθμό των εμφανίσεών τους και σε περίπτωση που κάποιος από αυτούς έχει συμμετάσχει σε αγώνα ως Πρώτος Διαιτητής, φαίνονται ο αριθμός των κίτρινων, των δεύτερων κίτρινων και των κόκκινων καρτών που έχει δώσει καθ' όλη τη διάρκεια του πρωταθλήματος.

Table	Referees			
Matches	Name	Appearances	Yellow Cards	Second Yellow Cards
Players	Chris Kavanagh	31	65	1
Teams	Dan Cook	33	-	-
Referees	Sian Massey-Ellis	24	-	-
Managers	Kevin Friend	40	59	1
Stadiums	Jonathan Moss	40	65	2
More Stats	Marc Perry	31	-	-
Exit	Daniel Robathan	29	-	-
	Simon Hooper	34	29	-
	Michael Oliver	33	85	2
	Stuart Burt	34	-	-
	Simon Bennett	34	-	-
	Mike Dean	40	92	2
	Stuart Attwell	31	87	1
	Constantine Hatzidakis	33	-	-
	Neil Davies	29	-	-
	Darren England	33	35	-
	Anthony Taylor	48	80	1
	Gary Bewick	34	-	-
	Adam Nunn	37	-	-
	Peter Banks	29	51	1

Εικόνα 29: Διαιτητές πρωταθλήματος

Πατώντας το κουμπί Managers, εκτελείται η συνάρτηση buttonPushed6() όπου ο χρήστης μπορεί να δει τους προπονητές των ομάδων του πρωταθλήματος. Στο διπλανό Frame φαίνονται τόσο οι προπονητές οι οποίοι έχουν ακόμη συμβόλαιο με την ομάδα, καθώς και αυτοί που έχουν απολυθεί μέσα στη σεζόν (2020-21) του πρωταθλήματος. Οι πρώτοι φαίνονται στην «αρχική σελίδα» του Frame, ενώ πατώντας το κουμπί Released Referees, φαίνονται οι απολυμένοι προπονητές (δίνεται η δυνατότητα επιστροφής στην αρχική σελίδα μέσω του κουμπιού Return).

The screenshot shows a window titled "Premier League 2020-2021". On the left is a vertical menu bar with the following items: Table, Matches, Players, Teams, Referees, Managers, Stadiums, More Stats, and Exit. The "Managers" item is currently selected. To the right of the menu is a table titled "Managers" with the following columns: Name, Team, Start of Contract, and Possible end of Contract. The table lists 22 managers, each associated with a specific team and their contract details. At the bottom of the table are two buttons: "Released Managers" and "Update/Insert".

Managers			
Name	Team	Start of Contract	Possible end of Contract
Brendan Rodgers	Leicester City	2019-02-27	None
David Moyes	West Ham United	2019-12-29	None
Graham Potter	Brighton & Hove Albion	2019-05-20	None
Jürgen Klopp	Liverpool	2015-10-08	None
Marcelo Bielsa	Leeds United	2018-06-15	None
Mikel Arteta	Arsenal	2019-12-22	None
Pep Guardiola	Manchester City	2016-07-01	None
Ralph Hasenhüttl	Southampton	2018-12-06	None
Sean Dyche	Burnley	2012-10-30	None
Thomas Tuchel	Chelsea	2021-01-26	None
Carlo Ancelotti	Everton	2019-12-22	2021-06-30
Nuno Espírito	Wolverhampton Wanderers	2017-05-31	2021-06-30
Paul Heckingbottom	Sheffield United	2021-03-13	2021-06-30
Roy Hodgson	Crystal Palace	2017-09-12	2021-06-30
Ryan Mason	Tottenham Hotspur	2021-04-19	2021-06-30
Sam Allardyce	West Bromwich Albion	2020-12-16	2021-06-30
Scott Parker	Fulham	2019-02-28	2021-06-30
Steve Bruce	Newcastle United	2019-07-17	2021-10-20
Dean Smith	Aston Villa	2018-10-10	2021-11-07
Ole Gunnar	Manchester United	2018-12-19	2021-11-21

Εικόνα 30: Προπονητές Πρωταθλήματος

Ακόμη δίνεται και η δυνατότητα τροποποίησής τους και ενσωμάτωσης καινούργιων προπονητών.

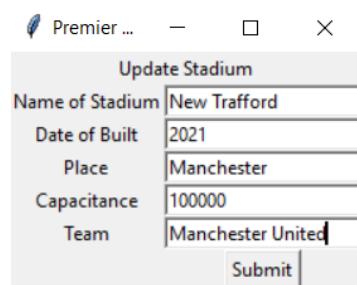
Πατώντας το κουμπί Stadiums, εκτελείται η συνάρτηση buttonPushed7() όπου ο χρήστης μπορεί να δει τα γήπεδα που χρησιμοποιήθηκαν ως έδρες των ομάδων του πρωταθλήματος καθώς και διάφορες πληροφορίες τους, όπως η χωρητικότητα, η ημερομηνία κατασκευής, η τοποθεσία κατασκευής και η ομάδα στην οποία είναι έδρα.

The screenshot shows a window titled "Premier League 2020-2021". On the left is a vertical menu bar with the following items: Table, Matches, Players, Teams, Referees, Managers, Stadiums, More Stats, and Exit. The "Stadiums" item is currently selected. To the right of the menu is a table titled "Stadiums" with the following columns: Name, Date of Built, Place, Capacitance, and Team. The table lists 20 stadiums, each with its name, construction date, location, capacity, and the team it belongs to. At the bottom of the table are two buttons: "Update/Insert" and "Released Managers".

Stadiums				
Name	Date of Built	Place	Capacitance	Team
Etihad Stadium	2003	Manchester	55097	Manchester City
New Trafford	2021	Manchester	100000	Manchester United
Anfield	1894	Liverpool	54074	Liverpool
Stamford Bridge	1877	London	41837	Chelsea
King Power Stadium	2002	Leicester	32500	Leicester City
London Stadium	2011	London	60000	West Ham United
Tottenham Hotspur Stadium	2018	London	62062	Tottenham Hotspur
Emirates Stadium	2006	London	60704	Arsenal
Elland Road	1898	Leeds	37890	Leeds United
Goodison Park	1892	Liverpool	40157	Everton
Villa Park	1897	Birmingham	42788	Aston Villa
St James' Park	1880	Newcastle	52409	Newcastle United
Molineux Stadium	1889	Wolverhampton	31700	Wolverhampton Wanderers
Selhurst Park	1924	London	26255	Crystal Palace
Saint Mary's	2001	Southampton	32689	Southampton
Amex Stadium	2011	Brighton	30750	Brighton & Hove Albion
Turf Moor	1993	Burnley	22546	Burnley
Craven Cottage	1896	London	25700	Fulham
The Hawthorns	1900	West Bromwich	26445	West Bromwich Albion
Bramall Lane	1855	Sheffield	33000	Sheffield United

Εικόνα 31: Στάδια Ομάδων του Πρωταθλήματος

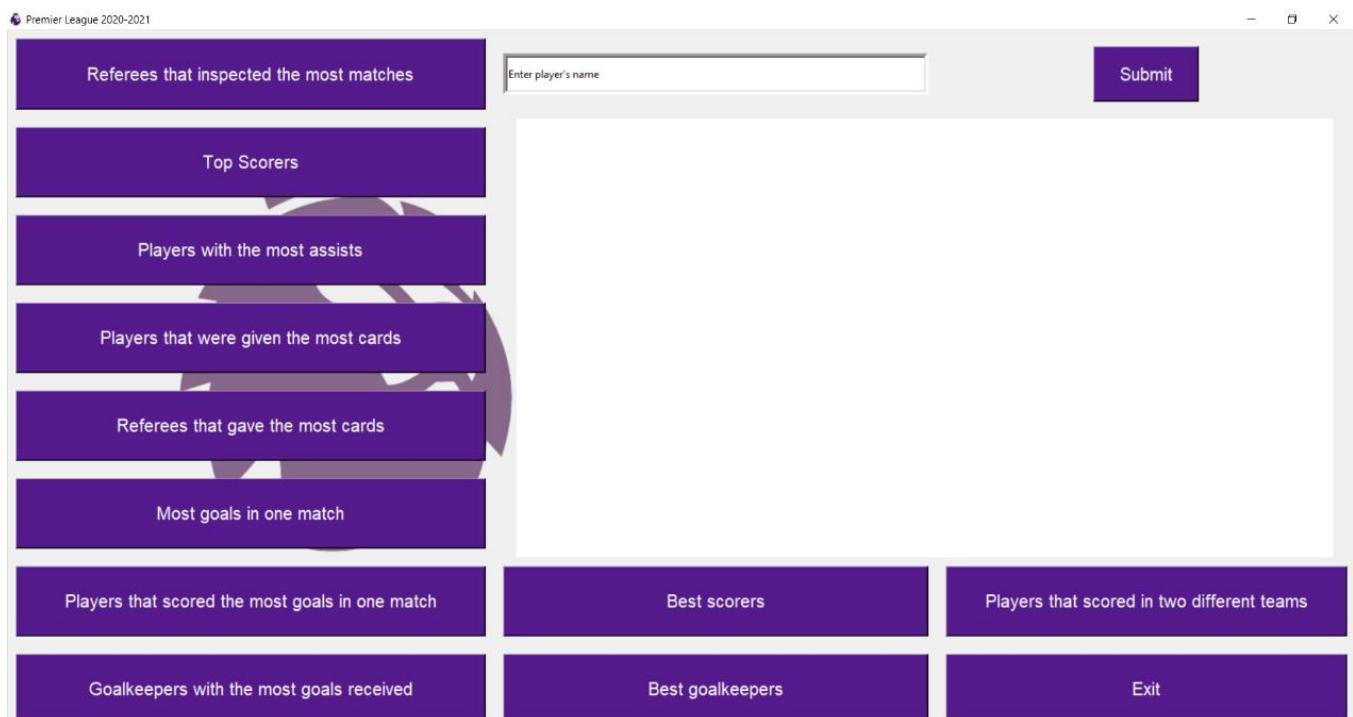
Ακόμη δίνεται και η δυνατότητα τροποποίησής τους και ενσωμάτωσης καινούργιων γηπέδων. Όπως φαίνεται παρακάτω:



Εικόνα 32: Ενσωμάτωση νέου γηπέδου

Καλό θα ήταν μετά από κάθε Update/Insert ο χρήστης να ξανατρέχει τον κώδικα της εφαρμογής, ώστε αυτές οι αλλαγές να φαίνονται και γραφικά στην εφαρμογή μας.

Η δεύτερη σελίδα της εφαρμογής που εμφανίζεται όταν ο χρήστης πατάει το κουμπί More Stats της αρχικής σελίδας φαίνεται στην παρακάτω εικόνα:



Εικόνα 33: Δεύτερη οθόνη εφαρμογής

Αν πατηθεί το κουμπί Referees that inspected the most matches , το οποίο το δημιουργούμε με τον εξής κώδικα:

```
self.b1 = tk.Button (self.root1,text='Referees that inspected the most matches', width=50, height=3, fg='#FFFFFF', bg='#551A8B', font=12, command=self.buttonPushed9)

self.b1.grid(row=0, column=0, padx=10, pady=10, sticky='ns')

τότε εκτελείται η συνάρτηση self.buttonPushed9() και εμφανίζεται στον λευκό καμβά αυτό που φαίνεται στην παρακάτω εικόνα, δηλαδή οι διαιτητές που έχουν διαιτητεύσει τους περισσότερους αγώνες καθώς και τον αριθμό των εμφανίσεών τους:
```

The screenshot shows a Tkinter application window. At the top left is a text input field labeled "Enter player's name". To its right is a purple "Submit" button. Below these is a title bar with the text "REFEREES WITH THE MOST INSPECTIONS PREMIER LEAGUE 2020-2021". Underneath is a table with a purple header row. The header columns are "First Name", "Last Name", and "Inspections". The data rows show the following information:

First Name	Last Name	Inspections
Anthony	Taylor	48
Andy	Madley	44
Robert	Jones	42
Mike	Dean	40
Jonathan	Moss	40
Kevin	Friend	40
Craig	Pawson	39
Martin	Atkinson	39
Graham	Scott	37
Adam	Nunn	37

Εικόνα 34: Διαιτητές με τις περισσότερες εμφανίσεις

Αν πατηθεί το κουμπί Top Scorers , το οποίο το δημιουργούμε με τον εξής κώδικα:

```
self.b2=tk.Button(self.root1,text='Top Scorers', width=50,height=3,fg='#FFFFFF',bg='#551A8B',font=12,command=self.buttonPushed10)

self.b2.grid(row=1, column=0, padx=10, pady=10, sticky='ns')

τότε εκτελείται η συνάρτηση self.buttonPushed10() και εμφανίζεται στον λευκό καμβά αυτό που φαίνεται στην παρακάτω εικόνα, δηλαδή οι παίκτες που σκόραραν τα περισσότερα γκολ κατά τη διάρκεια του πρωταθλήματος καθώς και τα γκολ που έβαλαν:
```

<input type="text" value="Enter player's name"/>	<input type="button" value="Submit"/>																																	
TOP SCORERS PREMIER LEAGUE 2020-2021																																		
How many players do you want to be shown:	10																																	
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="background-color: #800080; color: white;">First Name</th><th style="background-color: #800080; color: white;">Last Name</th><th style="background-color: #800080; color: white;">Goals scored</th></tr> </thead> <tbody> <tr><td>Harry</td><td>Kane</td><td>23</td></tr> <tr><td>Mohamed</td><td>Salah</td><td>22</td></tr> <tr><td>Bruno</td><td>Fernandes</td><td>18</td></tr> <tr><td>Patrick</td><td>Bamford</td><td>17</td></tr> <tr><td>Son</td><td>Heung-min</td><td>17</td></tr> <tr><td>Dominic</td><td>Calvert-Lewin</td><td>16</td></tr> <tr><td>Jamie</td><td>Vardy</td><td>15</td></tr> <tr><td>Ollie</td><td>Watkins</td><td>14</td></tr> <tr><td>Alexandre</td><td>Lacazette</td><td>13</td></tr> <tr><td>İlkay</td><td>Gündoğan</td><td>13</td></tr> </tbody> </table>		First Name	Last Name	Goals scored	Harry	Kane	23	Mohamed	Salah	22	Bruno	Fernandes	18	Patrick	Bamford	17	Son	Heung-min	17	Dominic	Calvert-Lewin	16	Jamie	Vardy	15	Ollie	Watkins	14	Alexandre	Lacazette	13	İlkay	Gündoğan	13
First Name	Last Name	Goals scored																																
Harry	Kane	23																																
Mohamed	Salah	22																																
Bruno	Fernandes	18																																
Patrick	Bamford	17																																
Son	Heung-min	17																																
Dominic	Calvert-Lewin	16																																
Jamie	Vardy	15																																
Ollie	Watkins	14																																
Alexandre	Lacazette	13																																
İlkay	Gündoğan	13																																

Εικόνα 35: Top Scorers Πρωταθλήματος

Αν πατηθεί το κουμπί Players with the most assists , το οποίο το δημιουργούμε με τον εξής κώδικα:

```
self.b3=tk.Button(root1,text='Players with the most assists',  
width=50,height=3,fg='#FFFFFF',bg='#551A8B',font=12,command=self.buttonPushed11)  
  
self.b3.grid(row=2,column=0,padx=10,pady=10,sticky='ns')
```

τότε εκτελείται η συνάρτηση self.buttonPushed11() και εμφανίζεται στον λευκό καμβά αυτό που φαίνεται στην παρακάτω εικόνα, δηλαδή εμφανίζονται οι παίκτες με τις περισσότερες assists καθώς και τον αριθμό των assist:

PLAYERS WITH THE MOST ASSISTS PREMIER LEAGUE 2020-2021																																		
How many players do you want to be shown:	10																																	
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="background-color: #800080; color: white;">First Name</th><th style="background-color: #800080; color: white;">Last Name</th><th style="background-color: #800080; color: white;">Total assists</th></tr> </thead> <tbody> <tr><td>Harry</td><td>Kane</td><td>14</td></tr> <tr><td>Bruno</td><td>Fernandes</td><td>12</td></tr> <tr><td>Kevin</td><td>De Bruyne</td><td>12</td></tr> <tr><td>Jack</td><td>Grealish</td><td>10</td></tr> <tr><td>Son</td><td>Heung-min</td><td>10</td></tr> <tr><td>Raphael</td><td>Dias Belloli</td><td>9</td></tr> <tr><td>Jamie</td><td>Vardy</td><td>9</td></tr> <tr><td>Marcus</td><td>Rashford</td><td>9</td></tr> <tr><td>Pascal</td><td>Groß</td><td>8</td></tr> <tr><td>Jack</td><td>Harrison</td><td>8</td></tr> </tbody> </table>		First Name	Last Name	Total assists	Harry	Kane	14	Bruno	Fernandes	12	Kevin	De Bruyne	12	Jack	Grealish	10	Son	Heung-min	10	Raphael	Dias Belloli	9	Jamie	Vardy	9	Marcus	Rashford	9	Pascal	Groß	8	Jack	Harrison	8
First Name	Last Name	Total assists																																
Harry	Kane	14																																
Bruno	Fernandes	12																																
Kevin	De Bruyne	12																																
Jack	Grealish	10																																
Son	Heung-min	10																																
Raphael	Dias Belloli	9																																
Jamie	Vardy	9																																
Marcus	Rashford	9																																
Pascal	Groß	8																																
Jack	Harrison	8																																

Εικόνα 36: Παίκτες με τις περισσότερες assist

Αν πατηθεί το κουμπί Players that were given the most cards, το οποίο το δημιουργούμε με τον εξής κώδικα:

```
self.b4=tk.Button(root1,text='Players that were given the most cards',width=50,height=3,fg='#FFFFFF',bg='#551A8B',font=12,command=self.buttonPushed12)
```

```
self.b4.grid(row=3,column=0,padx=10,pady=10,sticky='ns')
```

τότε εκτελείται η συνάρτηση self.buttonPushed12() και εμφανίζεται στον λευκό καμβά αυτό που φαίνεται στην παρακάτω εικόνα, δηλαδή οι παίκτες που δέχτηκαν τις περισσότερες κάρτες καθώς και τον αριθμό των καρτών:

The screenshot shows a Tkinter application window. At the top left is an input field labeled "Enter player's name". To its right is a purple "Submit" button. Below these is a title bar with the text "PLAYERS WITH THE MOST CARDS RECEIVED PREMIER LEAGUE 2020-2021". Underneath is a table with three columns: "First Name", "Last Name", and "Cards Received". A dropdown menu above the table indicates the number of rows to be shown, currently set to 10. The table lists 10 players:

First Name	Last Name	Cards Received
John	McGinn	12
Conor	Gallagher	11
Harry	Maguire	11
Douglas	Luiz	10
Kalvin	Phillips	10
John	Lundstram	9
Yves	Bissouma	9
Luka	Milivojević	9
Mason	Holgate	9
Pierre	Højbjerg	9

Εικόνα 37: Παίκτες με τις περισσότερες κάρτες

Αν πατηθεί το κουμπί Referees that gave the most cards , το οποίο το δημιουργούμε με τον εξής κώδικα:

```
self.b5=tk.Button(root1,text='Referees that gave the most cards',width=50,height=3,fg='#FFFFFF',bg='#551A8B',font=12,command=self.buttonPushed13)
```

```
self.b5.grid(row=4,column=0,padx=10,pady=10,sticky='ns')
```

τότε εκτελείται η συνάρτηση self.buttonPushed13() και εμφανίζεται στον λευκό καμβά αυτό που φαίνεται στην παρακάτω εικόνα, δηλαδή οι διαιτητές που έδωσαν τις περισσότερες κάρτες καθώς και τον αριθμό των καρτών:

The screenshot shows a Tkinter application window. At the top left is an input field labeled "Enter player's name". To its right is a purple "Submit" button. Below these is a title bar with the text "REFEREES THAT GAVE THE MOST CARDS PREMIER LEAGUE 2020-2021". Underneath is a table with three columns: "First Name", "Last Name", and "Cards". A dropdown menu above the table indicates the number of rows to be shown, currently set to 10. The table lists 10 referees:

First Name	Last Name	Cards
Mike	Dean	100
Stuart	Attwell	90
Michael	Oliver	88
Paul	Tierney	86
Anthony	Taylor	84
Craig	Pawson	83
David	Coote	71
Jonathan	Moss	67
Chris	Kavanagh	67
Kevin	Friend	62

Εικόνα 38: Διαιτητές που έδωσαν τις περισσότερες κόκκινες κάρτες

Αν πατηθεί το κουμπί Most goals in one match, το οποίο το δημιουργούμε με τον εξής κώδικα:

```
self.b6=tk.Button(root1,text='Most      goals      in      one  
match',width=50,height=3,fg='#FFFFFF',bg='#551A8B',font=12,command=se  
lf.buttonPushed14)  
  
self.b6.grid(row=5,column=0,padx=10,pady=10,sticky='ns')
```

τότε εκτελείται η συνάρτηση self.buttonPushed14() και εμφανίζεται στον λευκό καμβά αυτό που φαίνεται στην παρακάτω εικόνα, δηλαδή οι αγώνες στους οποίους μπήκαν τα περισσότερα γκολ, καθώς και πόσα ήταν αυτά και σε ποια αγωνιστική ήταν ο αγώνας:



The screenshot shows a Tkinter window with a purple header bar containing the text "MATCHES WITH THE MOST GOALS PREMIER LEAGUE 2020-2021". Below the header is a dropdown menu labeled "How many matches do you want to be shown:" with the value set to "10". To the right of the dropdown is a purple "Submit" button. The main area contains a table with the following data:

Team1 vs Team2	Score	matchweek	Total goals
Aston Villa vs Liverpool	7-2	4	9
Manchester United vs Southampton	9-0	22	9
Manchester United vs Leeds United	6-2	14	8
Liverpool vs Leeds United	4-3	1	7
Everton vs West Bromwich Albion	5-2	2	7
Leeds United vs Fulham	4-3	2	7
Southampton vs Tottenham Hotspur	2-5	2	7
Manchester City vs Leicester City	2-5	3	7
Manchester United vs Tottenham Hotspur	1-6	4	7
Aston Villa vs Southampton	3-4	7	7

Εικόνα 39: Αγώνες με τα περισσότερα γκολ

SQL που εκτελείται για να προκύψει το αποτέλεσμα:

```
SELECT IIF(E.home,T1.name,T2.name) AS team1,IIF(E.home!=1,T1.name,T2.name) AS  
team2,score,matchweek,(CAST(SUBSTR(score, 1, 1) AS INTEGER)+CAST(SUBSTR(score, 3, 3)  
AS INTEGER)) AS total_goals  
  
FROM (enters AS E1 LEFT JOIN (SELECT match_ID,team_ID AS team_ID2,home AS home2 FROM  
enters) AS E2 ON E1.match_ID=E2.match_ID AND E1.team_ID<>E2.team_ID2) AS E,Match_ AS  
M,Team AS T1,Team AS T2  
  
WHERE E.match_ID=M.match_ID AND T1.team_ID=E.team_ID AND T2.team_ID=E.team_ID2  
GROUP BY E.match_ID  
ORDER BY total_goals DESC  
LIMIT {};
```

Αν πατηθεί το κουμπί Goalkeepers with the most goals received, το οποίο το δημιουργούμε με τον εξής κώδικα:

```
self.b8=tk.Button(root1,text='Goalkeepers with the most goals received',width=50,height=3,fg='#FFFFFF',bg='#551A8B',font=12,command=self.buttonPushed16)
```

```
self.b8.grid(row=7,column=0,padx=10,pady=10,sticky='ns')
```

τότε εκτελείται η συνάρτηση self.buttonPushed16() και εμφανίζεται στον λευκό καμβά αυτό που φαίνεται στην παρακάτω εικόνα, δηλαδή εμφανίζονται οι τερματοφύλακες στους οποίους έχουν βάλει τα περισσότερα γκολ και ο αριθμός των γκολ:

The screenshot shows a Python application window with a light gray background. At the top left is a white input field with a black border containing the placeholder text "Enter player's name". To its right is a dark purple rectangular button with the word "Submit" in white. Below this is a purple header bar with the text "GOALKEEPERS THAT RECEIVED THE MOST GOALS PREMIER LEAGUE 2020-2021" in white. Underneath the header is a form with a dropdown menu labeled "How many goalkeepers do you want to be shown:" containing the number "10". A table follows, with a purple header row containing the columns "First Name", "Last Name", and "Goals Received". The data in the table is as follows:

First Name	Last Name	Goals Received
Sam	Johnstone	74
Vicente	Guaita	64
Aaron	Ramsdale	62
Alex	McCarthy	54
Illan	Meslier	52
Rui	Patrício	51
Alphonse	Areola	48
Kasper	Schmeichel	46
Emiliano	Martinez	44
Łukasz	Fabiański	44

Εικόνα 40: Τερματοφύλακες που έχουν δεχθεί τα περισσότερα γκολ

Αν πατηθεί το κουμπί Players that scored the most goals in one match, το οποίο το δημιουργούμε με τον εξής κώδικα:

```
self.b7=tk.Button(root1,text='Players that scored the most goals in one match', width=50, height=3, fg='#FFFFFF', bg='#551A8B', font=12, command=self.buttonPushed15)
```

```
self.b7.grid(row=6,column=0,padx=10,pady=10,sticky='ns')
```

τότε εκτελείται η συνάρτηση self.buttonPushed15() και εμφανίζεται στον λευκό καμβά αυτό που φαίνεται στην παρακάτω εικόνα, δηλαδή εμφανίζεται το ονοματεπώνυμο του παίκτη, ο αγώνας στον οποίο ο παίκτης σκόραρε τα περισσότερα γκολ, καθώς και πόσα ήταν αυτά και σε ποια αγωνιστική ήταν ο αγώνας:

The screenshot shows a web-based application for searching player statistics. At the top left is a text input field labeled "Enter player's name". To its right is a purple "Submit" button. Below this is a purple header bar with the text "THE MOST GOALS SCORED BY A PLAYER IN ONE MATCH PREMIER LEAGUE 2020-2021". Underneath the header is a dropdown menu labeled "How many players do you want to be shown" with a value of "10" selected. The main content area is a table with the following columns: Team 1 vs Team 2, Score, Matchweek, First name, Last name, Goals scored, and Team of the player. The table lists ten matches from the 2020-2021 Premier League season where a single player scored the most goals.

Team 1 vs Team 2	Score	Matchweek	First name	Last name	Goals scored	Team of the player
Southampton vs Tottenham Hotspur	2–5	2	Son	Heung-min	4	Tottenham Hotspur
Wolverhampton Wanderers vs Burnley	0–4	33	Chris	Wood	3	Burnley
Aston Villa vs Liverpool	7–2	4	Ollie	Watkins	3	Aston Villa
Everton vs West Bromwich Albion	5–2	2	Dominic	Calvert-Lewin	3	Everton
Aston Villa vs Leeds United	0–3	6	Patrick	Bamford	3	Leeds United
Arsenal vs Leeds United	4–2	24	Pierre-Emerick	Aubameyang	3	Arsenal
Tottenham Hotspur vs Sheffield United	4–0	34	Gareth	Bale	3	Tottenham Hotspur
Leicester City vs Sheffield United	5–0	28	Kelechi	Iheanacho	3	Leicester City
Manchester City vs Leicester City	2–5	3	Jamie	Vardy	3	Leicester City
Liverpool vs Leeds United	4–3	1	Mohamed	Salah	3	Liverpool

Εικόνα 41: Παίκτες με τα περισσότερα γκολ σε έναν αγώνα

SQL που εκτελείται για να προκύψει το αποτέλεσμα:

```

SELECT IIF(E.home,T1.name,T2.name) AS team1,IIF(E.home!=1,T1.name,T2.name) AS
team2,score,matchweek,first_name,last_name,max(goals) as
goals,IIF(T1.team_ID=B.team_ID,T1.name,T2.name) AS his_team

FROM participates AS P1, Player AS P2,(enters AS E1 LEFT JOIN (SELECT match_ID,team_ID
AS team_ID2,home AS home2 FROM enters) AS E2 ON E1.match_ID=E2.match_ID AND
E1.team_ID<>E2.team_ID2) AS E,Team AS T1,Team AS T2,Match_ AS M,belongs AS B

WHERE P1.player_ID=P2.player_ID AND position<>'GK' AND T1.team_ID=E.team_ID AND
T2.team_ID=E.team_ID2 AND E.match_ID=M.match_ID AND P1.match_ID=E.match_ID AND
B.player_ID=P1.player_ID

GROUP BY P1.player_ID

ORDER BY goals DESC

LIMIT {};

```

Αν πατηθεί το κουμπί Best Scorers , το οποίο το δημιουργούμε με τον εξής κώδικα:

```
self.b11=tk.Button(root1,text='Best scorers',width=50,height=3,fg='#FFFFFF',bg='#551A8B',font=12,command=self.buttonPushed17)

self.b11.grid(row=6,column=1,padx=10,pady=10,sticky='ns')
```

τότε εκτελείται η συνάρτηση self.buttonPushed17() και εμφανίζεται στον λευκό καμβά αυτό που φαίνεται στην παρακάτω εικόνα, δηλαδή εμφανίζει το ονοματεπώνυμο του παίκτη που τα λεπτά ανά γκολ που σκοράρει είναι από τα λιγότερα:

First Name	Last Name	Minutes per goal
Gareth	Bale	83
Kelechi	Iheanacho	121
Diogo	Jota	123
Harry	Kane	134
Edinson	Cavani	137
Sergio	Agüero	139
Mohamed	Salah	139
Alexandre	Lacazette	147
Joe	Willock	152
Anthony	Elanga	155

Εικόνα 42: Καλύτεροι scorer

Αν πατηθεί το κουμπί Best goalkeepers, το οποίο το δημιουργούμε με τον εξής κώδικα:

```
self.b12=tk.Button(root1,text='Best goalkeepers', width=50, height=3,
fg='#FFFFFF',bg='#551A8B',font=12,command=self.buttonPushed18)

self.b12.grid(row=7,column=1,padx=10,pady=10,sticky='ns')
```

τότε εκτελείται η συνάρτηση self.buttonPushed18() και εμφανίζεται στον λευκό καμβά αυτό που φαίνεται στην παρακάτω εικόνα, δηλαδή εμφανίζει το ονοματεπώνυμο των τερματοφύλακα που τα λεπτά ανά γκολ που δέχεται είναι από τα μεγαλύτερα:

First Name	Last Name	Minutes per Goal Received
Caoimhín	Kelleher	180
Kiko	Casilla	135
Edouard	Mendy	112
None	Ederson	112
Robert	Sánchez	93
None	Alisson	92
Darren	Randolph	90
Zack	Steffen	90
Bernd	Leno	82
Dean	Henderson	80

Εικόνα 43: Καλύτεροι τερματοφύλακες

SQL που εκτελείται για να προκύψει το αποτέλεσμα:

```
SELECT
first_name, last_name, sum(participation_in_match)/sum(IIF(home!=1, (CAST(SUBSTR(score,
1, 1) AS INTEGER)), (CAST(SUBSTR(score, 3, 3) AS INTEGER)))) AS
minutes_per_goal_received

FROM participates AS P1, Player AS P2, Match_ AS M, enters AS E, belongs AS B

WHERE position=='GK' AND M.match_ID=P1.match_ID AND M.match_ID=E.match_ID AND
B.team_ID=E.team_ID AND B.player_ID=P1.player_ID AND P1.player_ID=P2.player_ID

GROUP BY P1.player_ID

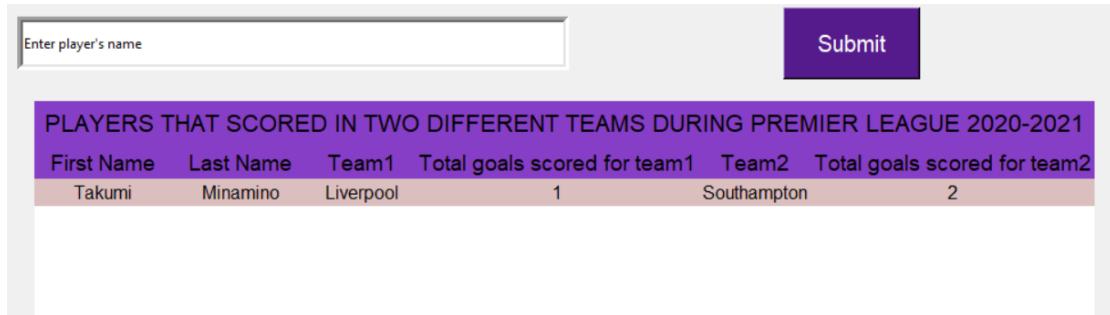
ORDER BY minutes_per_goal_received DESC

LIMIT {};
```

Αν πατηθεί το κουμπί Players that scored in two different teams, το οποίο το δημιουργούμε με τον εξής κώδικα:

```
self.b13=tk.Button(root1,text='Players that scored in two different
teams',width=50,height=3,fg='#FFFFFF',bg='#551A8B',font=12,command=se
lf.buttonPushed19)
self.b13.grid(row=6,column=2,columnspan=2,padx=10,pady=10,sticky='ns'
)
```

τότε εκτελείται η συνάρτηση self.buttonPushed19() και εμφανίζεται στον λευκό καμβά αυτό που φαίνεται στην παρακάτω εικόνα, δηλαδή εμφανίζεται το ονοματεπώνυμο του παίκτη που σκόραρε σε δύο διαφορετικές ομάδες, καθώς και σε ποιες ομάδες έχει συμμετέχει ακριβώς και πόσα γκολ έβαλε στην καθεμία:



Εικόνα 44: Παίκτες σκόραραν σε δύο διαφορετικές ομάδες την ίδια σεζόν

SQL που εκτελείται για να προκύψει το αποτέλεσμα:

```
CREATE VIEW IF NOT EXISTS players_that_changed_teams
AS
SELECT player_ID,team_ID,SUM(goals) AS total_goals
FROM (participates NATURAL JOIN Match_) NATURAL JOIN belongs
WHERE      strftime("%Y-%m-%d",datetime_)<contract_end_day      AND      strftime("%Y-%m-
%d",datetime_)>contract_start_day AND player_ID IN(
SELECT player_ID
FROM belongs
GROUP BY player_ID
```

```

HAVING COUNT(*)>1

) GROUP BY player_ID,team_ID;

SELECT first_name,last_name,name AS team1,total_goals AS total_goals_at_team1,name2 AS
team2,total_goals2 AS total_goals_at_team2

FROM (Player NATURAL JOIN players_that_changed_teams NATURAL JOIN(SELECT team_ID AS
team_ID2,player_ID,total_goals AS total_goals2 FROM players_that_changed_teams) NATURAL
JOIN Team) AS T1 JOIN (SELECT name AS name2,team_ID FROM Team) AS T2 ON
T1.team_ID2=T2.team_ID

WHERE T1.team_ID<>team_ID2 AND player_ID IN(
SELECT player_ID
FROM players_that_changed_teams
WHERE total_goals>0
GROUP BY player_ID
HAVING COUNT(*)>1
) LIMIT 1;

```

Αν πατηθεί το κουμπί Submit, το οποίο το δημιουργούμε με τον εξής κώδικα:

```

self.b10=tk.Button(root1,text='Submit',width=10,height=2,fg='#FFFFFF'
,bg='#551A8B',font=12,command=self.submit)

self.b10.grid(row=0,column=2,padx=10,pady=10)

```

αφότου έχει εισαχθεί από τον χρήστη το ονοματεπώνυμο του παίκτη που θέλει να αναζητήσει τότε εκτελείται η συνάρτηση self.submit() και εμφανίζεται στον λευκό καμβά αντό που φαίνεται στην παρακάτω εικόνα, δηλαδή εμφανίζεται το ονοματεπώνυμο του παίκτη με κάποια χαρακτηριστικά του και τα συνολικά του στατιστικά από τους αγώνες που συμμετείχε:

First Name	Last Name	Shirt Number	Height	Date of birth	Goals	Saves	Passes	Assists	Shots	Shots on target	Tackles
Harry	Kane	4	188	1993-07-28	23	None	657	14	134	47	18

Εικόνα 45: Εύρεση Χαρακτηριστικών και Στατιστικών Παίκτη

Ενδεικτικό παράδειγμα SQL κώδικα για update:

Στο συγκεκριμένο απόσπασμα κάνοντας update τα στατιστικά ενός παίκτη σε έναν αγώνα, εφόσον ο παίκτης αυτός υπάρχει στην ομάδα και παίζει στον συγκεκριμένο αγώνα γίνεται η τροποποίηση με βάσει τα δεδομένα που βάζει ο χρήστης.

```
sql='''update participates set
start={},participation_in_match={},position="{}",shots={},shots_on_target={},assists={},
saves={},passes={},tackles={},end_of_participation=start_of_participation+participation_in_match
where player_ID={} and
match_ID={}''' .format(p[2],p[3],p[4],p[5],p[6],p[7],p[8],p[9],p[10],11[0][0],match_ID)
```

Σε περίπτωση που ο χρήστη βάλει στο input ‘;’, ‘DELETE’ ή ‘DROP’ με σκοπό να σβήσει τη βάση ή κάποιο πίνακα τότε εμφανίζεται ένα label που τον αναγκάζει να βάλει άλλα στοιχεία.

Ενδεικτικό παράδειγμα SQL κώδικα για insert:

Αν δεν υπάρχει ήδη το γήπεδο/στάδιο γίνεται insert με τον εξής κώδικα:

```
insert into Stadium(name,date_built,place,capacitance)
values("{}", "{}", "{}", "{}")''' .format(p[0],p[1],p[2],p[3])
```

Επιδόσεις Εφαρμογής:

Τρέχοντας τον κώδικα της εφαρμογής και πατώντας όλα τα κουμπιά που περιέχει, βλέπουμε στο IDLE Shell της Python, τους χρόνους που κάνουν να εκτελεστούν τα ερωτήματα της SQL για το κάθε κομπί. Οπως φαίνεται παρακάτω, παρατηρούμε ότι οι χρόνοι αυτοί είναι αρκετά μικροί και σε αυτό συνέβαλαν τα indexes που χρησιμοποιήσαμε σε κάποιους μεγάλους πίνακες της βάσης μας.

The screenshot shows the Python IDLE Shell window with the title 'IDLE Shell 3.10.1'. The code area contains several SQL queries being executed against a database named 'project_final'. The output shows the time taken for each query to execute, such as '0.01051 sec' for a SELECT count(*) query and '0.00019 sec' for an INSERT INTO Stadium query. The queries cover various database operations like selecting from tables like Coach, Player, Match, Team, and Stadium, and performing calculations like sum(participations) and sum(goals). The interface includes standard Python file operations like 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help' menus.

Βιβλιογραφία:

- [1] <https://fbref.com>
- [2] https://en.wikipedia.org/wiki/list_of_premier_league_managers
- [3] <https://www.worldfootball.net/>
- [4] <https://www.premierleague.com/>

Για την προετοιμασία της εφαρμογής και τη συλλογή των δεδομένων είδαμε τα παρακάτω βίντεο:

- [1] <https://youtu.be/XVv6mJpFOb0>
- [2] <https://youtu.be/SEQjNEawceo>
- [3] <https://youtu.be/YXPyB4XeYLA>

Παράτημα:

Οδηγίες Εγκατάστασης:

Για συλλογή δεδομένων μέσω web-scraping:

(1) Σε περίπτωση που δεν είναι εγκατεστημένες οι βιβλιοθήκες pandas, beautifulsoup4, requests, lxml και xlswriter, πρέπει να γίνει η εγκατάστασή μέσω του terminal(γραμμή εντολών) του υπολογιστή. Αυτό γίνεται μέσω της εντολής pip install <όνομα της βιβλιοθήκης>. Περισσότερες πληροφορίες για αυτές παρέχονται στο documentation της python.

(2) Για να τρέξουμε τα αρχεία του web-scraping για τη συλλογή των δεδομένων των ιστοσελίδων:

Αρχικά, τρέχουμε πρώτα τα αρχεία κώδικα με ονόματα matches, stadiums, players, belongs, coaches, Team. Από αυτά τα αρχεία προκύπτουν οι πίνακες των αγώνων(Match_), των σταδίων(Stadium), των παικτών(Player), των συμβολαίων των παικτών(belongs), των προπονητών(Coach) και των ομάδων(Team) αντίστοιχα. Από το αρχείο matches.py προκύπτει επίσης και ένα επιπλέον αρχείο excel με όνομα matches2, το οποίο χρησιμοποιείται σε επόμενα αρχεία κώδικα και το οποίο περιέχει επιπλέον τις ομάδες που πήραν μέρος σε κάθε αγώνα καθώς και το λινκ του κάθε αγώνα. Τα αρχεία κώδικα που χρειάζονται αυτό το αρχείο excel είναι τα enters.py, attributesViolation.py, substitutes.py, referee_inspects.py και το participates.py, τα οποία πρέπει να βρίσκονται όλα στον ίδιο φάκελο. Τρέχοντάς τα, προκύπτουν excel αρχεία με δεδομένα για τους πίνακες enters, attributesViolation, substitutes, referee, inspects και participates.

Λόγω του ότι ορισμένα δεδομένα από την ιστοσελίδα δεν ήταν σωστά, κάνουμε ορισμένες αλλαγές:

- Διαγράφουμε από το αρχεία excel Players και participates τους παίκτες με player_ID 172, 331, 370, 409, 458, 463, 495, 538, 559, 596, 601. Αυτούς του διαγράφουμε διότι είναι παίκτες που εμφανίζονται σε δύο ομάδες και άρα έχουν δύο IDs.
- Από το participates excel ακόμη, διαγράφουμε στον παίκτη με player_ID 148 διαγράφουμε τα στατιστικά του από τον αγώνα με match_ID 363, διότι είχε κενά στατιστικά και δημιουργούσε πρόβλημα στη βάση μας.

Ακόμη, επειδή θεωρήσαμε ότι σε κάθε αγώνα δεν έχουμε επιπλέον καθυστερήσεις (αφού δεν υπήρχε σαν δεδομένο στην ιστοσελίδα), φτιάχνουμε τους χρόνους συμμετοχής των παικτών στους αγώνες, τους χρόνους που γίνονται οι αλλαγές και τους χρόνους που δίνονται κάρτες, ώστε σε κάθε αγώνα και για κάθε ομάδα που συμμετέχει σε αυτόν, οι παίκτες της να έχουν χρόνο συμμετοχής το πολύ 990 λεπτά, τρέχουμε το αρχείο fix_times_new. Αυτό το αρχείο κάνει αυτή την τροποποίηση και δημιουργεί τα τελικά αρχεία δεδομένων για τους πίνακες participates, substitutes και attributesViolation. Το συγκεκριμένο αρχείο κώδικα πρέπει να βρίσκεται επίσης μαζί με τα υπόλοιπα αρχεία excel που φτιάξαμε.

Επιπλέον, διαγράφουμε από το τελικό αρχείο excel του participates τους εξής παίκτες:

-player_ID = 355 στο match_ID = 130 και εκεί που έχει χρόνο συμμετοχής 64 λεπτά

-player_ID = 33 στο match_ID = 208 και εκεί που έχει χρόνο συμμετοχής 87 λεπτά

-player_ID = 383 στο match_ID = 206 και εκεί που έχει χρόνο συμμετοχής 47 λεπτά

Αυτούς επίσης τους βγάζουμε και από το attributes_violation. Τους συγκεκριμένους παίκτες τους βγάζουμε διότι η ιστοσελίδα από όπου πήραμε τα δεδομένα μας, θεωρούσε τις εκτελέσεις πέναλτι, τις αποκρούσεις πέναλτι, τις ασσίστες και τα αυτογκόλ ως καινούργια συμμετοχή ή ως κάρτες.

Τέλος προσθέτουμε στο αρχείο excel participates, έναν παίκτη με τα εξής χαρακτηριστικά:

player_ID = 368, match_ID= 217 , start = Y , goals = 0, participation_in_match=79, saves = NULL, position = MF , start_of_participation =0 και end_of_participation = 79, ενώ τα άλλα στοιχεία είναι 0.

Ο λόγος που το κάνουμε αυτό είναι γιατί ο συγκεκριμένος βγήκε από τον αγώνα ως τραυματίας και δεν αντικαταστάθηκε από κάποιον άλλο παίκτη.

Έχοντας, τρέξει όλα αυτά τα αρχεία προκύπτουν τα αρχεία excel με τα δεδομένα της βάσης μας τα οποία και πρέπει να αποθηκευτούν σε φάκελο με όνομα excel_csv (Αυτά τα αρχεία excel υπάρχουν έτοιμα και στο φάκελο με όνομα excel_csv).

(3) Δημιουργία Βάσης: Για τη δημιουργία της βάσης μας, εκτελούμε τον κώδικα της python με όνομα initialize_database. Αυτό το αρχείο δημιουργεί τη βάση δεδομένων μας και ενσωματώνει τα δεδομένα των αρχείων excel και csv που πήραμε από το προηγούμενο βήμα σε αυτή, ενώ ακόμη ενσωματώνει και τα δεδομένα υπολογιζόμενων γνωρισμάτων. Προκειμένου να είναι δυνατή η εκτέλεση του αρχείου αυτού, πρέπει να είναι εγκατεστημένες οι βιβλιοθήκες pandas,sqlite3 και time της python. Ο τρόπος εγκατάστασής τους είναι ο ίδιος με αυτό που είπαμε και στο προηγούμενο βήμα. Η βάση δεδομένων που προκύπτει υπάρχει στον παραδιδόμενο φάκελο με όνομα project το οποίο και μπορεί να ανοίξει κανείς μέσω της εφαρμογής της SQLite κάνοντας open database και επιλέγοντας το συγκεκριμένο αρχείο. Για την σωστή λειτουργία του προγράμματος αυτού, θα πρέπει ο χώρος στον οποίο βρίσκεται να περιλαμβάνει οπωσδήποτε τον φάκελο excel_csv. Τα αρχεία αυτά excel τα διαβάζουμε μέσω της εντολής pd.read_<τύπος_αρχείου>(r'excel_csv\< όνομα_αρχείου>.<τύπος αρχείου>').

(4) Εκτέλεση Εφαρμογής: Η εκτέλεση της εφαρμογής γίνεται εκτελώντας τον κώδικα της python με όνομα database_final. Συνιστούμε το τρέξιμο του κώδικα στο ίδιο χώρο με το αρχείο της βάσης και με τον φάκελο με όνομα images, διότι χρησιμοποιούμε εικόνες στην εφαρμογή μας που υπάρχουν σε αυτό το φάκελο. Προκειμένου να είναι δυνατή η εκτέλεση του αρχείου αυτού, πρέπει να είναι εγκατεστημένες οι βιβλιοθήκες tkinter, sqlite3 και time της python.

Παραδείγματα Χρήσης Προγράμματος:

Η εφαρμογή αυτή μπορεί να χρησιμοποιηθεί για διάφορους σκοπούς, όπως είναι η καταγραφή των δεδομένων ενός ποδοσφαιρικού πρωταθλήματος, καταγραφή των αγώνων καθώς και των στατιστικών των παικτών, ενώ ακόμη μπορεί να γίνει χρήση ώστε να δει διάφορα στατιστικά τα οποία είναι συνήθως αυτά που αναζητά κάποιος που ασχολείται με ένα ποδοσφαιρικό πρωτάθλημα.