**Football Database for a Professional Football Consulting Company**

**Team**

118010220 Haotian Ma (马浩天)

118010224 Yu Mao (毛宇)

118010335 Wei Wu (吴畏)

118010416 Shiqi Zhang (张诗琪)

**Abstact**

In professional football, there are a number of consulting companies that provide decision support for clubs and players. To help achieve their business goals, we designed and implemented a database that stores both latest and historical information of players, clubs, and leagues. This database supports a set of daily operations such as adding, updating, deleting, and searching for a player and/or a club. Furthermore, we analyzed and visualized the data for several selected practical problems.

**Contribution:**

|  |  |  |
| --- | --- | --- |
| Name | Student ID | Contribution |
| Haotian Ma (马浩天) | 118010220 | Database Implementation |
| Yu Mao (毛宇) | 118010224 | Diagram Production, Database Implementation, Performance Optimization |
| Wei Wu (吴畏) | 118010335 | Database Specification, Database Design, Data Analysis |
| Shiqi Zhang (张诗琪) | 118010416 | Database Operations, User Interface |

***NB*** All members contribute equally to this project.

**Database Specification:**

1. Requirements of the Organization:

a. Record information of professional football players.

i. Demographic information: Player ID, Name, Date of Birth, and Nationality.

ii. Anthropometric measurements: Height and Weight.

iii. Club & transfer information: Club, Loaned From, Wage, Value, Release Clause, Contract Valid Until, Team Number.

iv. Technical data: Overall, Potential, Position, Positional Rating, Preferred Foot, Weak Foot Rating, Skill Moves Rating, Tag, Pace, Shooting, Passing, Dribbling, Defending, Physic, GK Diving, GK Handling, GK Kicking, GK reflexes, GK Speed, GK Positioning.

b. Record information of professional football clubs: Club and League.

c. Maintain historical data.

d. Maintain the database.

i. Add / update / delete a player / club.

e. Analyze the information of players and clubs.

i. Analyze the players in terms of demographic information, anthropometric measurements, club & transfer information, and technical data.

ii. Analyze the clubs in terms of squad rating, squad value, squad wage.

iii. Provide career strategies for players.

iv. Provide operational strategies for clubs.

2. Constraints, assumptions, and properties:

a. In a single season, a player must be either goalkeeper or non-goalkeeper (general player).

b. In a single season, a player plays for at most one club. (Although in reality, a player can play for multiple clubs in a single season by transferring and/or loaning, we record only the club to which a player belongs at the beginning of each season. In other words, we update the database once a year.)

c. In a single season, a player may be loaned from at most one club. In other words, a player can belong to at most one club. If a player is not loaned, he belongs to the club he plays for.

d. In a single season, a club belongs to at most one league.

e. A league always belongs to a certain country/region.

f. In a single season, a player can belong to at most one country/region.

g. According to FIFA’s regulations, a player can represent at most one country/region at any given time. Although some players may have double nationalities, they can only register one at FIFA. However, they may change their nationality if they fulfill the requirements. Hence, a player may have different nationalities in different seasons.

h. A player’s name may change.

i. A player may have multiple (at least one) best positions.

j. In a single season, a player has exactly one rating for one position.

k. In a single season, a player may have multiple (including zero) tags.

l. A player’s overall should be dependent on other attributes. However, this functional dependency is not made public by EA Sports. Therefore, we just t reat overall as independent of other attributes.

m. A league’s name and tier may change over time. E.g. English Championship changed from tier 1 to tier 2.

n. A player’s potential must be greater than or equal to his overall in a single season.

o. A player’s preferred foot must be either left or right in a single season.

p. A player’s weak foot rating and skill move rating range from 1 to 5.

q. A player’s team number ranges from 0 to 99.

r. In a certain season, a player’s contract end year must be greater than the season’s end year. E.g. Contract Valid Until = 2023, Season = 20.

s. A player’s ratings (overall, pace, shooting, passing dribbling, defending, physic, gk\_diving, gk\_handling, gk\_kicking, gk\_reflexes, gk\_speed, gk\_positioning, and all positional ratings) range from 1 to 99.

t. Players’ height is in centimeters while weight is in kilograms. Wage, value, and release clause are in euros.

**Database Design and Implementation:**

1. Uninitialized entities:

Initially, the raw data is divided into four strong entities by their properties. The four entities are player, club, league and country/region. Those four entities are independent entities that should be generated. The primary keys are player’s id and season to determinate a unique player in a unique FIFA season; club’s id to determinate a unique club in the whole world; league’s id to determinate to a unique league in the whole world; region’s name or country’s name to determinate a unique nationality in the world.

1. Attributes of entities:

In player, it contains the basic information of players include:

player’s id, season, player’s name, date of birth, nationality, height, weight, club id, loaned from, wage, value, release clause, contract valid until, team number, overall, potential, position, positional rating, preferred foot, weak foot rating, skill moves rating, tag.

In club, it contains the basic information of the club: Club’s id, season, club’s name, league.

In league, it contains the basic information of the league: league’s id, season, league’s name, tier, country/region.

In country/region, it contains the basic information of the country or region: country/region’s name

1. Relations:

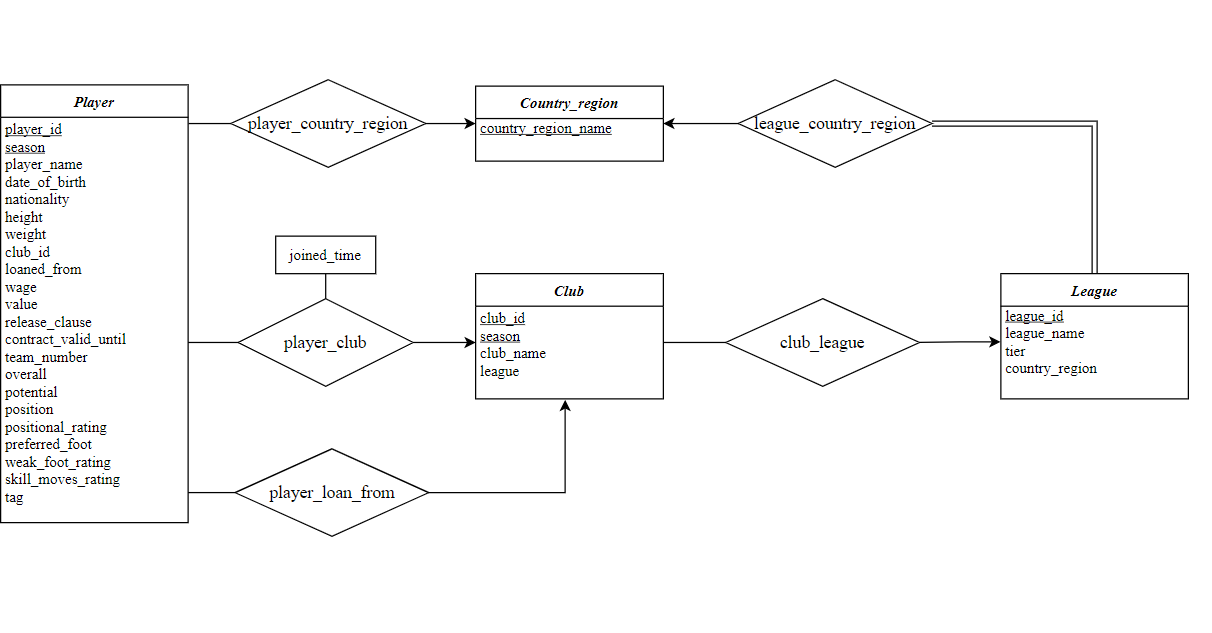
There exist some relations between entities. A player belongs to a unique club in a season, it is the many-to-one relation. A club must belong to a league, it is a many-to-one relation. A player has a unique nationality that is in a country or region, also a many-to-one relation. The uninitialized E-R diagram is generalized based on this business rule.

Figure 1.3.1: Uninitialized E-R Diagram

1. Normalization:

In the present schema, it is in the first normal form as it contains some partial dependencies and transitive dependency. Most of the attributes depend on season and player’s id, however the date of birth of the player has no relationship with the season. This is the partial dependency. And attribute position rating depends on position which is not a primary key. Therefore, a normalization is applied. After the normalization, the database is in third normal form.

Steps of Normalization:

FD&MVD:(

Player\_id, season 🡪player\_name, nationality, height, weight, club\_id, loaned\_from, wage, value, release\_clause, contract\_valid\_until, team\_number, overall, potential, position, positional rating, preferred foot, weak foot rating, skill\_moves\_rating, tag

Player\_id 🡪date\_of\_birth

Position🡪positional\_rating)

R1= (Position, player\_id, season, positional\_rating)

Remove component positional\_rating in table player

R2= (player\_id, date\_of\_birth)

Remove component date\_of\_birth in table player

R3= (Player\_id, season, tag)

Tag and position are entities:

R4= (tag)

R5= (position)

R6= (player\_id, season, player\_name, nationality, height, weight, club\_id, loaned\_from, wage, value, release\_clause, contract\_valid\_until, team\_number, overall, potential, preferred foot, weak foot rating, skill\_moves\_rating)

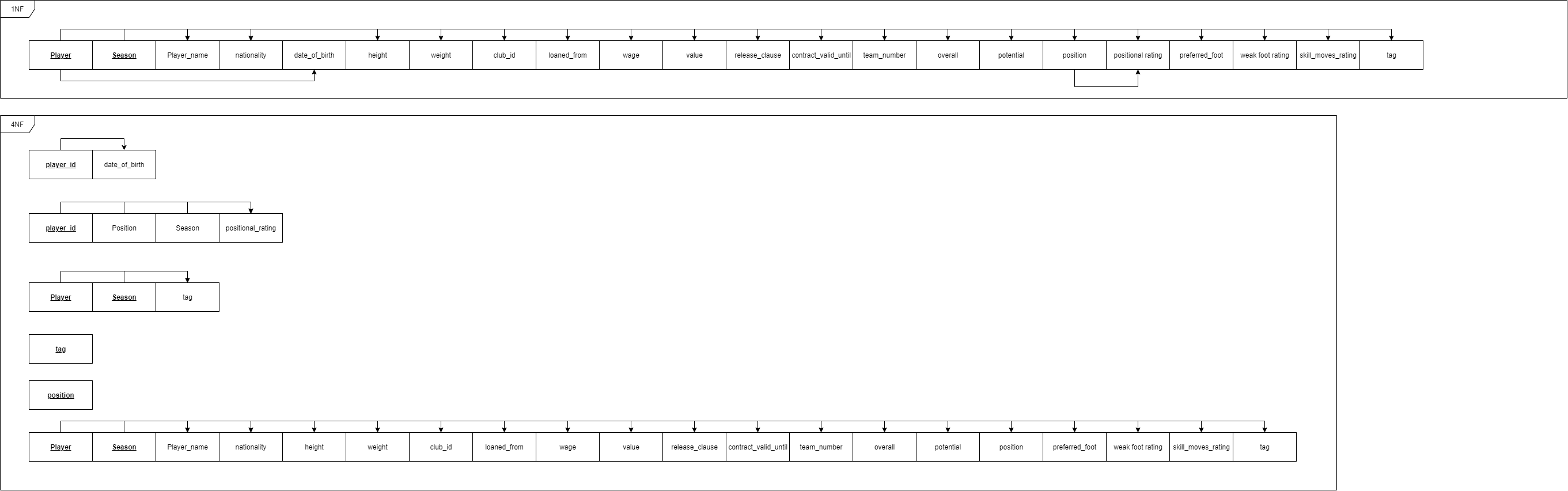


Figure 1.4.1: Normalization to 3NF

After normalization, we can get our final ERD:

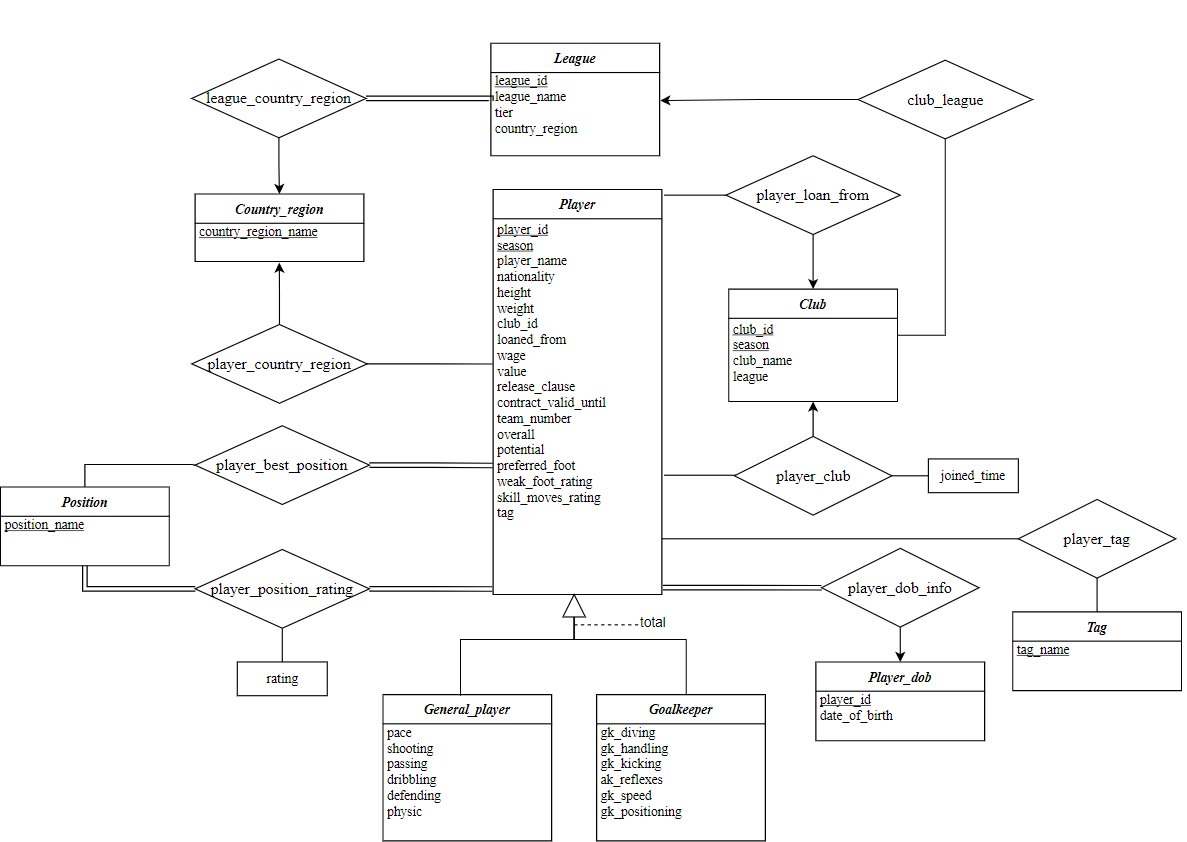


Figure 1.4.2: Normalized ERD

Then we need to reduce them to schemas for constructing tables in database:

***goalkeeper***(player\_id, season, gk\_diving, gk\_handling, gk\_kicking, gk\_reflexes, gk\_speed, gk\_positioning)

FK: (player\_id, season) from (player.player\_id, player.season)

FD: player\_id, season🡪gk\_diving, gk\_handling, gk\_kicking, gk\_reflexes, gk\_speed, gk\_positioning

MVD:None

***positions***(position\_name, position\_class)

FK: None

FD: position\_name🡪position\_class

MVD: None

***tag***(tag\_name)

FK: None

FD: None

MVD: None

***player\_best\_position***(player\_id, season, position\_name)

FK: (player\_id, season) from (player.player\_id, player.seaspn)

position\_name from positions.position\_name

FD: None

MVD: None

***player\_positional\_rating***(player\_id, season, rating, position\_name)

FK: (player\_id, season) from (player.player\_id, player.seaspn)

position\_name from positions.position\_name

FD: player\_id, season, position\_name🡪rating

MVD: None

***player\_tag***(player\_id, season, tag\_name)

FK: (player\_id, season) from (player.player\_id, player.seaspn)

tag\_name from tag.tag \_name

FD: None

MVD: None

***player***(player\_id, season, player\_name, player\_short\_name, nationality, height, weight, club\_id, loaned\_from, wage\_eur, value\_eur, release\_clause, contract\_valid\_until, team\_number, overall, potential, preferred\_foot, weak\_foot\_rating, skill\_moves\_rating)

FK: club\_id from club.club\_id

nationality from country\_region.country\_region\_name

FD: player\_id, season🡪all other attributes

MVD: None

***player\_dob\_info***(player\_id, date\_of\_birth)

FK: player\_id from player.player\_id

FD: player\_id🡪date\_of\_birth

MVD: None

***country\_region***(country\_region\_name)

FK: None

FD: None

MVD: None

***league***(league\_id, season, league\_name, tier, country\_region\_name)

FK: country\_region\_name from country\_region.country\_region\_name

FD: league\_id, season🡪 league\_name, tier, country\_region\_name

MVD: None

***club***(club\_id, season, club\_name, league\_id)

FK: league\_id from league.league\_id

FD: club\_id. season🡪club\_name, league\_id

MVD: None

***general\_player***(player\_id, season, pace, shooting, passing, dribbling, defending, physic)

FK: (player\_id, season) from (player.player\_id, player.seaspn)

FD: player\_id, season🡪pace, shooting, passing, dribbling, defending, physic

MVD:None

**Data Insertion:**

1. Preprocessed the data:

Our project will use real data source from FIFA official dataset[[1]](#footnote-1) which contains all the players’ information from season 14/15 to 20/21. The dataset is given in the form of 7 csv files(which is put in the /FIFADB/Data/), each of them contains the whole information of each player in each season.

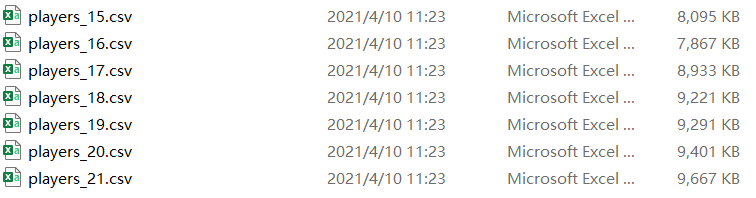


Figure 2.1.1: data source file

To make it more convenient to import data to the database, we first need to retrieve all the information needed for each table in the normalized schema. Thus we write a python program (/FIFADB/Codes/ step\_0\_process\_data.py) to scan through all the csv files and put data for each table in a corresponding csv file (put in the /FIFADB/Data/Data\_processed). Notice that we further clean and modify these data by modifying these files generated on python that is why the file generated from python program will be different from these files in the folder.

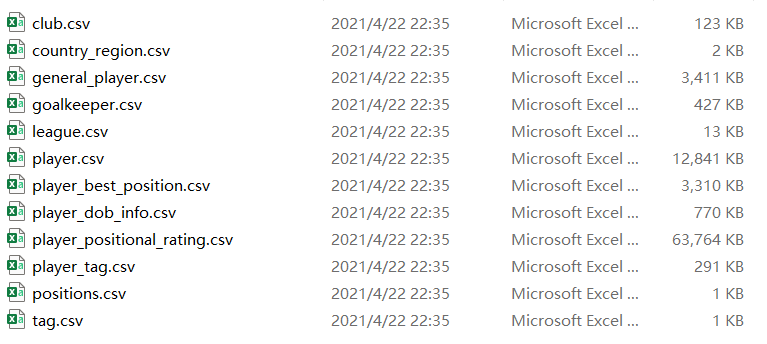


Figure 2.1.2: csv file for each table

1. Construct database tables and import those prepared data:

After preparing data, we then need to construct a database contains all the tables in normalized schema. Notice that to avoid any importing data abnormal, we construct the original table with all the attributes in type varchar, which we will set back to appropriate type in the later step. All the attributes are set in a proper size represented in length of varchar and we have set some attribute to be not null to follow our database rules. (All the codes of constructing tables are in

/FIFADB/Codes/ step\_1\_construct\_tables.sql).

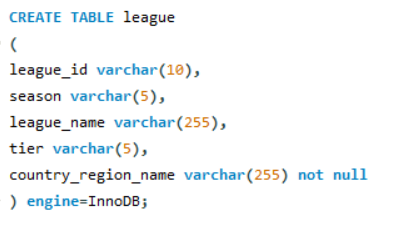


Figure 2.2.1: example of constructing tables

With all tables constructed, we then need to import all the csv file data into their tables. Codes used to import data are put in /FIFADB/Codes/step\_1\_construct\_tables.sql. Notice that if you want to execute this sql file, you need to change the csv files path to the corresponding data path in your computer.



Figure 2.2.2: example of importing data

1. Set keys and constraints of tables

In the next step, we need to set proper constraints to those tables. Firstly we need to clean some abnormal data which will cause failure of setting further constraints.



Figure 2.3.1: clean abnormal data

Since we initially constructing all the table using attribute type varchar, at this stage we need to convert them back to normal type.

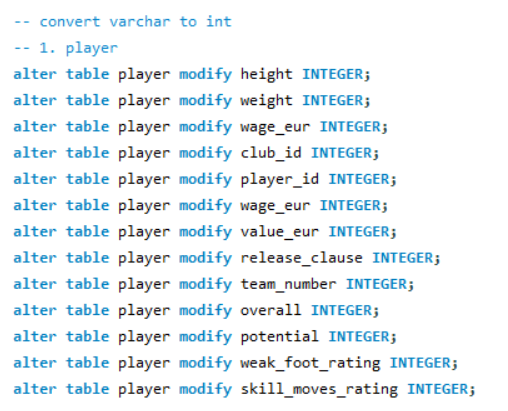


Figure 2.3.2: set attribute type(partial)

With all the attributes in proper types, we can then set primary keys and foreign keys for each table.

Notice that we need to first set up all the primary keys for all tables and set their foreign keys afterwards otherwise it will cause error.



Figure 2.3.3: set primary and foreign keys(partial)

To make sure the attributes in the database will follow the constraints we set, we also need to set check constraints for some attributes in certain table.

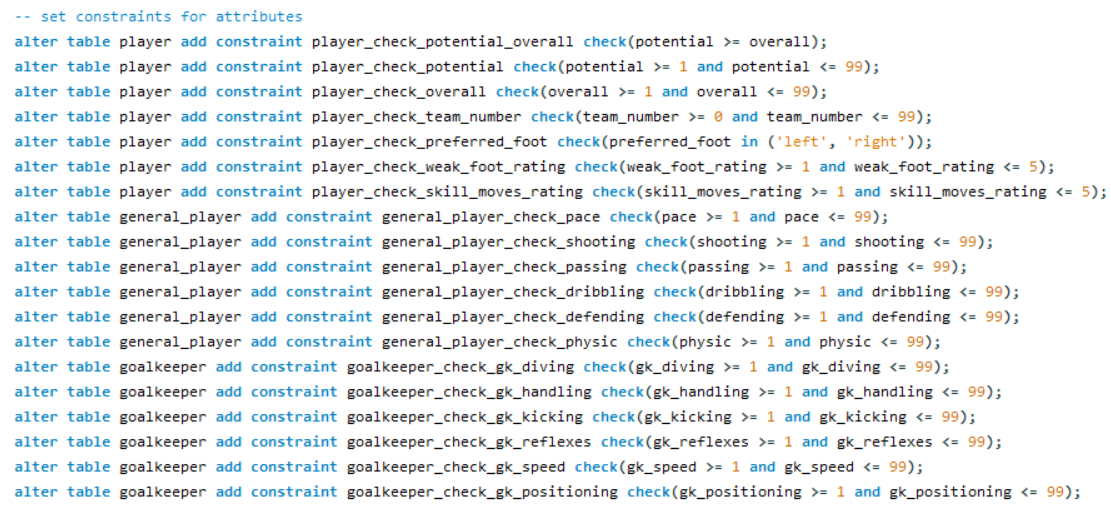


Figure 2.3.4: set check constraints for some attributes

Notice the codes mentioned in this section are put in /FIFADB/Codes/step\_3\_set\_constraints.sql.

**User Interface:**

We build up a website to simulate our database for practical usage. In practice, the database manager is the only one who can insert new data, so we create an Admin table to store the administer account information.

Pre-requirement

1. Packages

Werkzeug==1.0.1 / Flask\_Excel==0.0.7 / SQLAlchemy==1.3.24 / Flask\_Mail==0.9.1 / WTForms==2.3.1 / Flask\_WTF==0.14.3

XlsxWriter==1.2.7 / PyMySQL==0.9.2 / Flask==1.1.2 / Flask\_SQLAlchemy==2.4.3 / Flask\_Login==0.5.0 / flask\_bootstrap==3.3.7.1

1. Environment

*Python 3.9 / Chrome*

\*\*pip install -r requirements.txt\*\*”run this command for all package

In \*\*config.py\*\* to confige \*\*mysql\*\*'account and password

run \*\*views.py\*\*

ctrl + click on "\*\*http://127.0.0.1:5000.\*\*

1. Read the readme.md

4. Account & password

Administer account: admin / Password: 123456

Website functions

1. login

You can log in to get more authority as a administer (see figure 3.1.1)

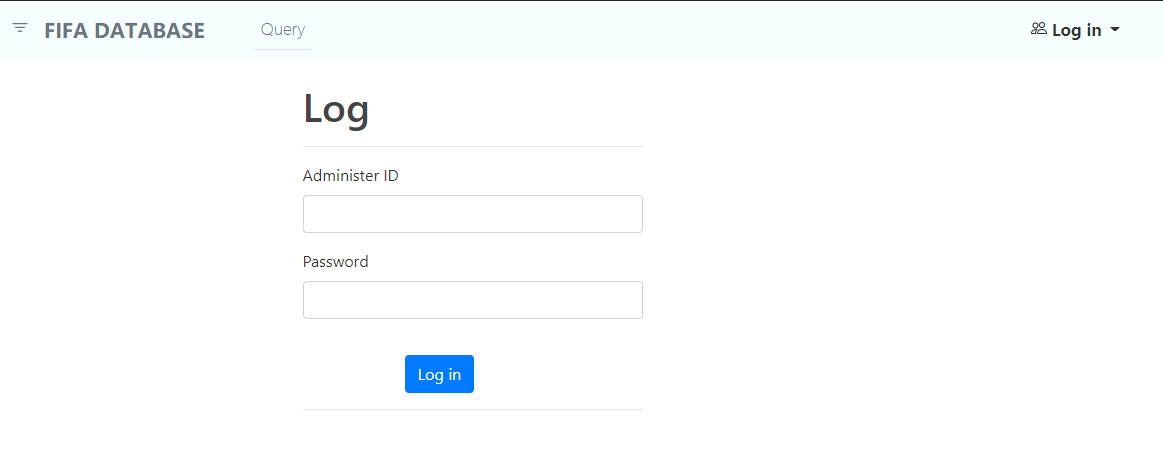
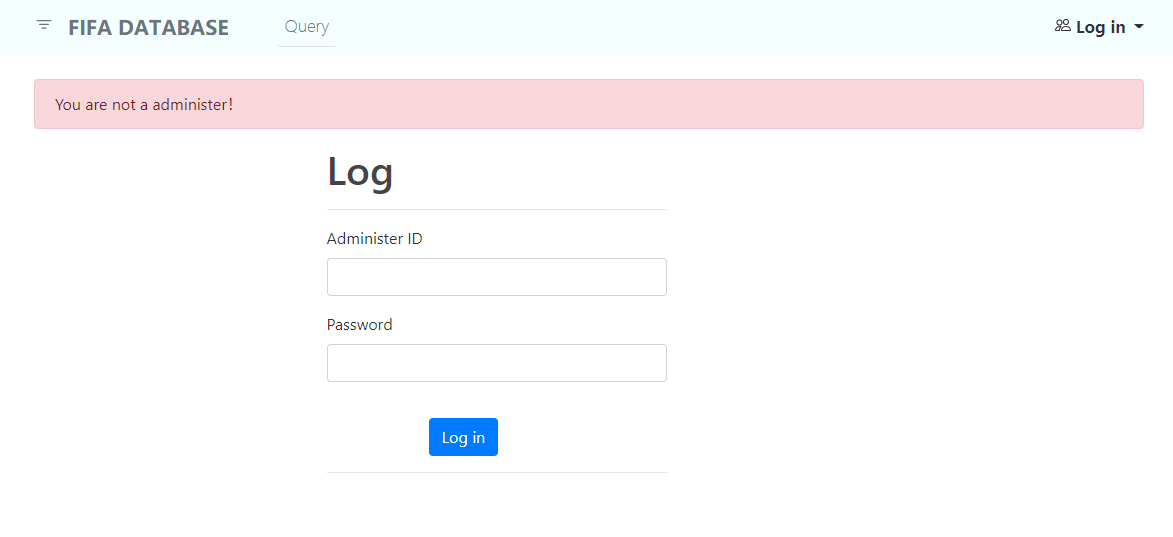
Otherwise, you will be warned when you try to register a new player.(see figure 3.1.2)

Figure 3.1.1

Figure 3.1.2

To ensure the correctness of the database, we will not allow the administrator front-end registration, of course, we will provide the administrator account and password to meet the requirements of the staff.

2. Administer Only

Only administer can insert new data.

\*The new player must have a unique id and season, we will check it if you insert duplicate information. (see figure 3.2.1)

\*The new player must have all information otherwise we will warn you about that.(see figure 3.2.2)

\*Once the information complete, the new\_player's information will be stored in the waiting list for further confirmation.(see figure 3.2.3)

\*Of course, the overall value should be guaranteed to be less than or equal to the potential. Our backstage will automatically detect the correctness of the input data to ensure the integrity of the database.

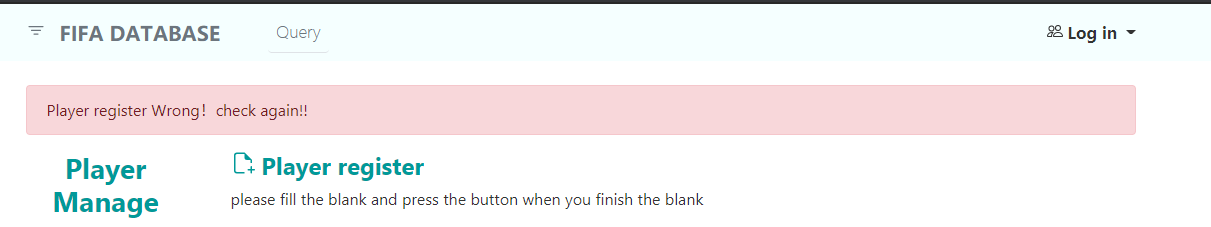
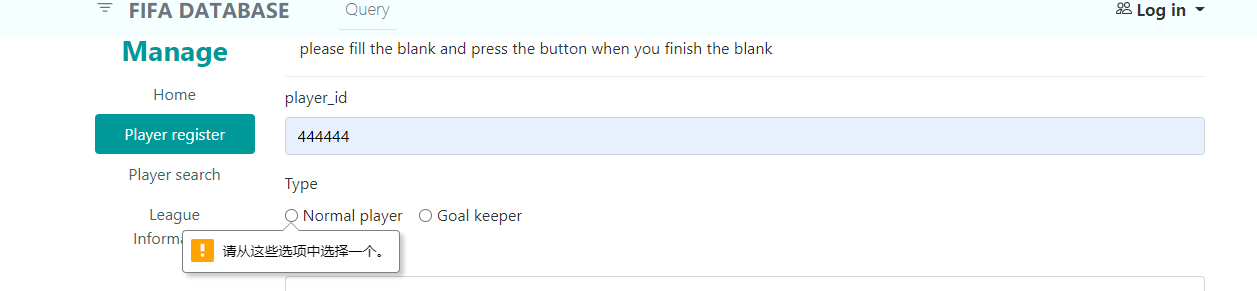


Figure 3.2.2

Figure 3.2.1

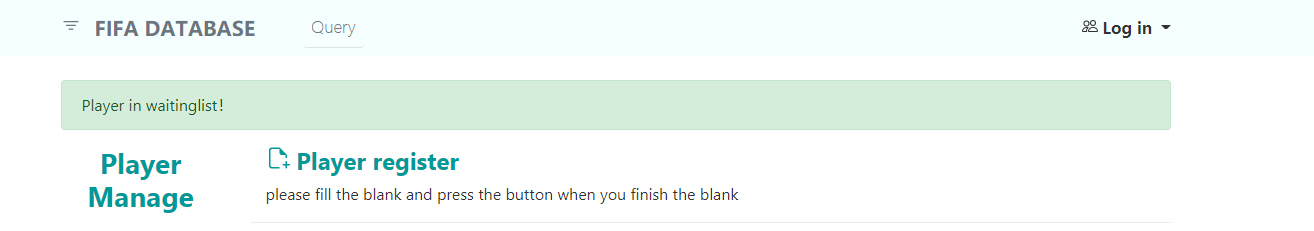


Figure 3.2.3

3. Functions administer and user both can use

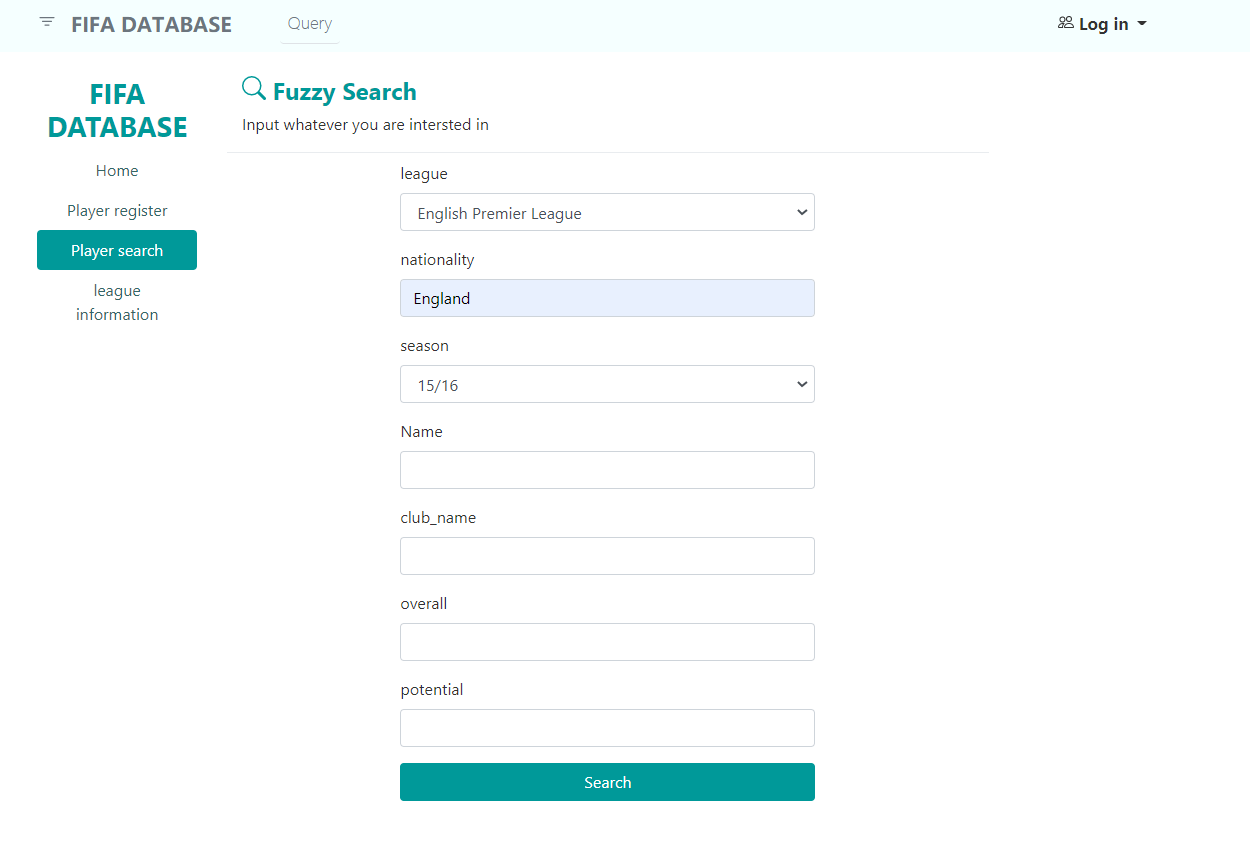
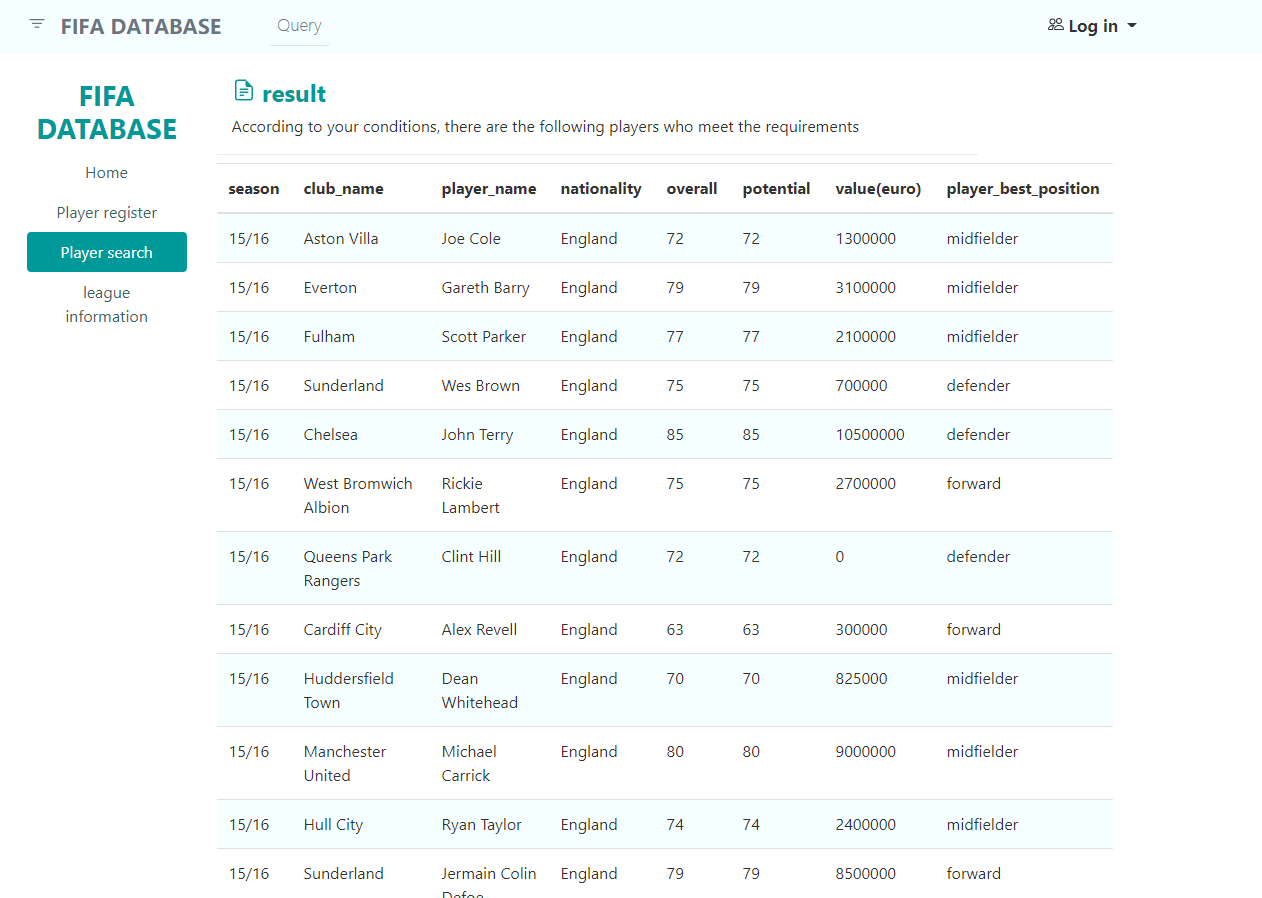
\*\*You can search players information when you get one of the search keys (Player\_id, season, player\_best\_position, league, club\_name, nationality, overall , potential)see figure(3.3.1) and you will get the information include (Player\_id, season, player\_best\_position, league, club\_name, nationality, overall , potential,value(euro)) .see figure(3.3.2)

Figure 3.3.1

Figure 3.3.2

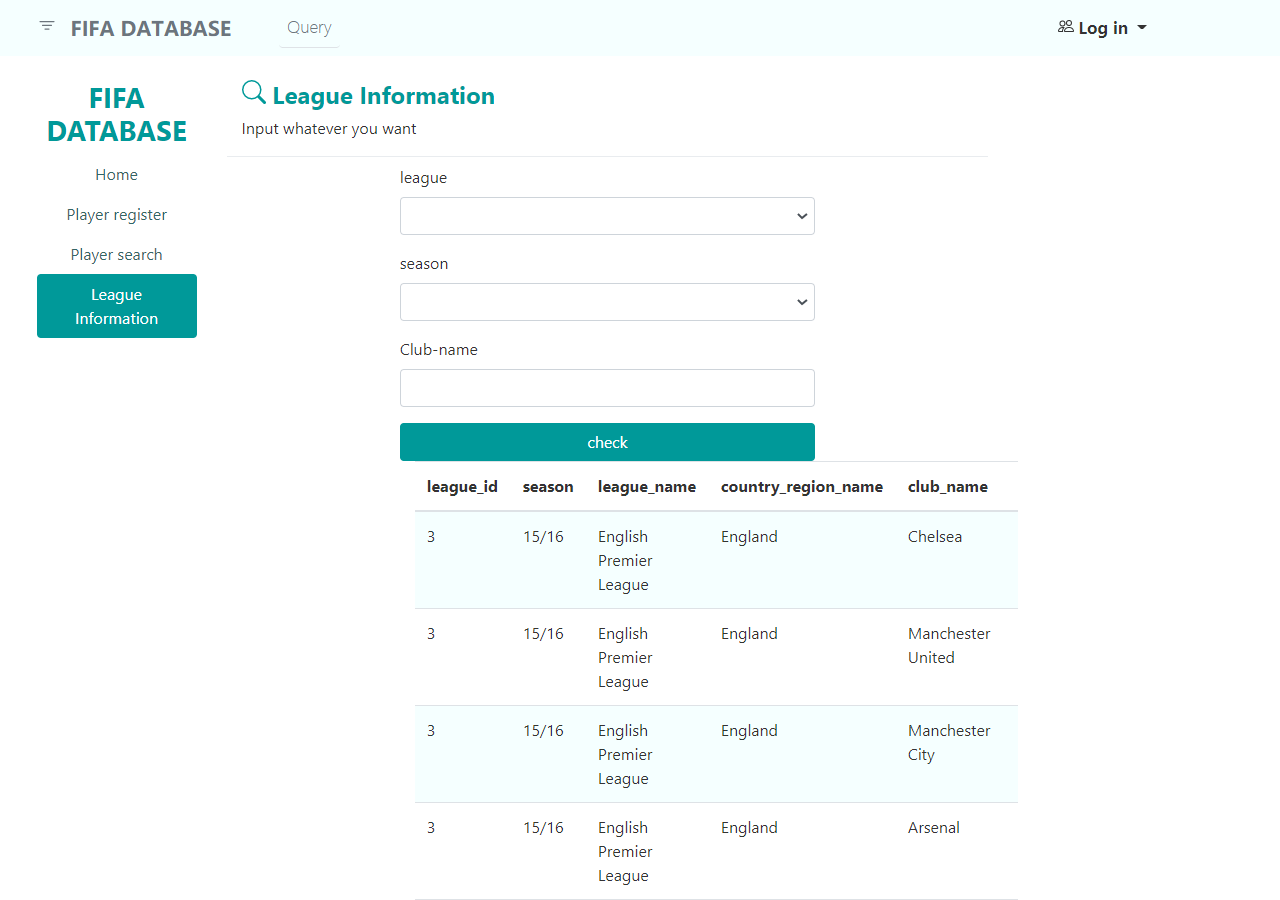
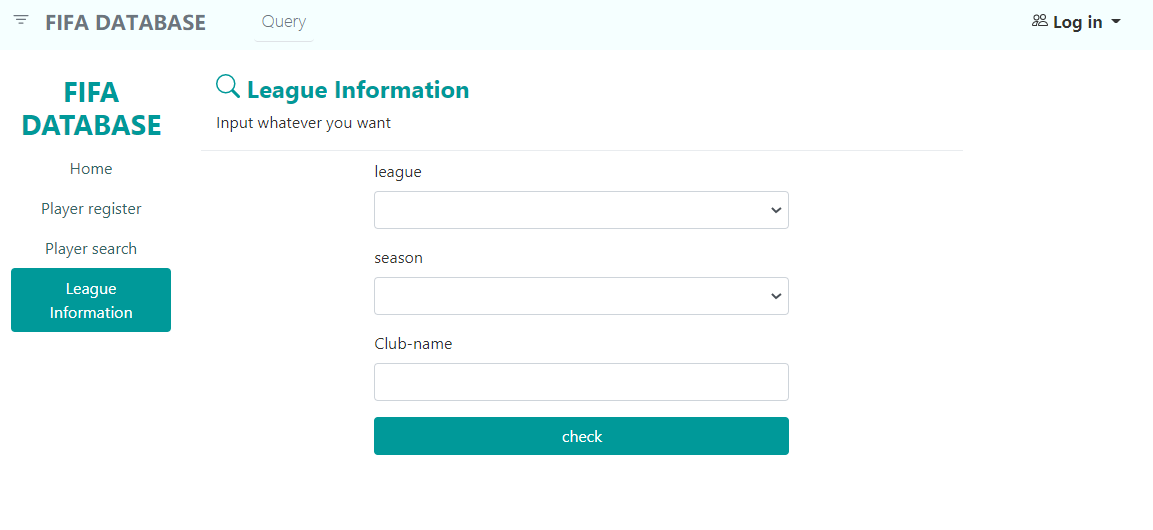
\*\*If you are not so familiar with the league and the club and their belongings, or you are not sure whether this club has been promoted in a certain season, then we have also prepared a page for you to get more information(see figure 3.3.3). You can enter whatever you want to know (league, club, season information) to get information. (see figure 3.3.4）

Figure 3.3.3

Figure 3.3.4

1. Sample SQL queries used for practical daily operations.

We produce queries to search player data,league data,insert a new player, delete a player and edit player’s information. You can try it directly through the website. The code will be attached behind,see in appendix code 1-5.

**Data Analysis:**

We inserted a large volume of actual data into our database schema. To illustrate that our database is capable of providing business intelligence, we raised a few practical questions.

1. How does the total salaries of the BIG 6 clubs of English Premiere League change from season 14/15 to 20/21?

a. Code Implementation

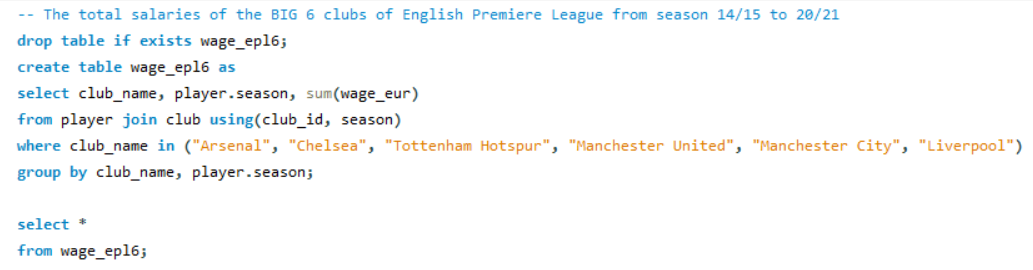


Figure 4.1.1

b. Query Output (see Appendix Table 1)

c. Data Visualization (see Appendix Figure A1)

d. Key Business Intelligence

From season 14/15 to 20/21, in general, Manchester City had the highest total wage; Tottenham Hotspur had the lowest total wage before Arsenal took its position in season 18/19.

From season 16/17 to 19/20, the total wage of Arsenal dropped significantly from about 3.1 million to 2 million.

From season 14/15 to 20/21, Manchester City, Liverpool, and Tottenham Hotspur generally had an upward trend in total wage.

1. The rank of clubs in each of the "Big Five" European football leagues in terms of the total wage in season 20/21.

a. Code Implementation



Figure 4.2.1

b. Query Output (see Appendix Table 2)

c. Data Visualization (see Appendix Figure A2-A8)

d. Key Business Intelligence

In season 20/21, clubs from English Premiere League had an average total wage that was much higher than clubs from other leagues.

In season 20/21, German 1. Bundesliga and French Ligue 1 had a relatively small standard deviation in total wages. While Spain Primera Division had the greatest standard deviation in total wages.

In English Premiere League, the highest total wage (Manchester City) was about only 6 times of the lowest one (Sheffield United). While in German 1. Bundesliga, the highest total wage (FC Bayern München) was also about 6 times of the lowest one (DSC Arminia Bielefeld).

In French Ligue 1, Paris Saint-Germain had a total wage that was approximately two times of the second highest one (AS Monaco) and 10 times of the lowest one (Dijon FCO).

In Italian Serie A, Inter had a total wage that was about 13 times of the lowest one (Crotone).

In Spanish La Liga, Real Madrid and FC Barcelona had very close total wages (nearly 5 million euros), which were also the highest among the clubs of the Big Five leagues. They were over 20 times higher than the lowest (Elche CF).

1. A Comparison of the Big 3 Clubs of La Liga in Terms of Overall Ratings of Different Positions from Season 18/19 to 20/21.

a. Code Implementation

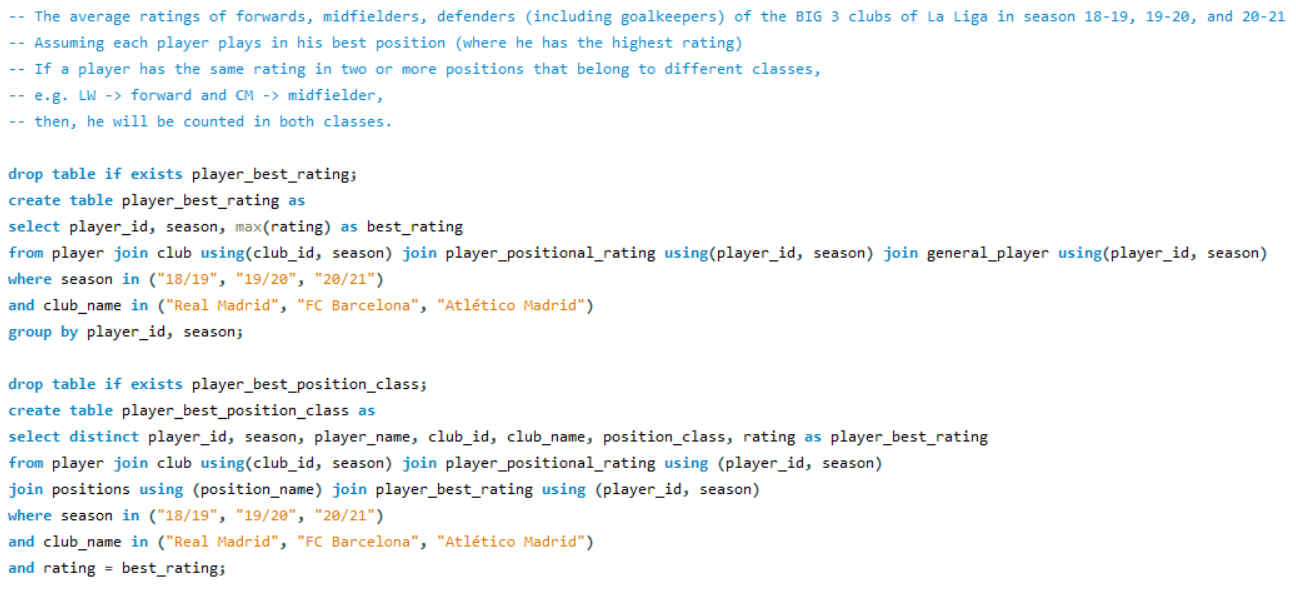




Figure 4.3.1

b. Query Output (See Appendix Table 3)

c. Data Visualization (See Appendix Figure A9 and A10)

d. Key Business Intelligence

During the three seasons, Read Madrid and FC Barcelona had close overall ratings, while Atlético Madrid had a lower overall rating.

In season 19/20 and 20/21, Atlético Madrid’s ratings of defenders and midfielders were not far behind those of FC Barcelona and Real Madrid. However, the front line of Atlético Madrid had a great disparity with the others.

The rating of defenders, midfielders as well as overall rating of Atlético Madrid showed an upward trend during the three seasons.

**Performance Optimization (by Indexes):**

Create Indexes on Data Fields

After detailed discussion, our team decide to set indexes (including hashing index) on attributes of four frequently used tables. Due to the fact that users usually search some conditions using different combination of attributes, we decide to set indexes on each attribute separately. All these attributes are listed in the following:

1. player:
   1. player\_id: hashing index. The number of player\_id is quite huge and player\_id distributes quite sparsely which makes them unsuitable for bit-map. Also the user usually will query the player\_id in using the exact search where hashing index will perform much better than B-tree.
   2. season: bit-map index. There are totally 7 seasons in this database which is relatively small size. Therefore a bit-map comparing with hashing and b-tree will be much better since it not only searches data fast and occupy little storage.
   3. player\_name: B-tree index. First, players’ name distributed sparsely which makes it unsuitable for bit-map since it will use a lot of bits to represent different values. Usually, user will not only search the player’s name by exact search but also by fuzzy search like searching with prefix. In this scenario, B-tree will perform much better than other types of indexes because of its tree structure allows efficient search for prefix search and exact search.
   4. player\_short\_name: B-tree index (same reason as player.player\_name).
   5. club\_id: hashing index. There are tons of club resulting lots of different club ids, which will make bit-map structure not sufficient in storage. Due to the fact that club\_id is usually looked up by exact search, where hashing will perform much better than B-tree theoretically.
2. club:
   1. club\_id: hashing index (same reason as player.club\_id).
   2. season: bit-map index (same reason as player.season)
   3. club\_name: B-tree index (same reason as player.player\_name)
   4. league\_id: bit-map index. The total number of league\_id is relatively small which makes it possible to use bit-map with sufficient storage. Also, most of the search of league\_id is using exact search, where bit-map performs quite well.
3. league:
   1. league\_id: bit-map index (same reason as club.league\_id).
   2. season: bit-map index (same reason as player.season)
   3. league\_name: B-tree index. User might use fuzzy search like using prefix search where B-tree will perform much better than other two types.
   4. country\_region\_name: B-tree index (same as league.league\_name).
4. player\_positional\_rating:
   1. player\_id: hashing index (same reason as player.player\_id).
   2. season: bit-map (same reason as player.season).
   3. positional\_name: bit-map (same reason as player.season).
5. player\_best\_position:
   1. player\_id: hashing index (same reason as player.player\_id).
   2. season: bit-map (same reason as player.season).
   3. position\_name: bit-map (same reason as player.season).

However, the case is a lit bit different in the implementation. After checking the MySQL documents, we find the MySQL does not support bit-map indexing, which means we need to transform the type of bit-map index to hashing or B-tree.

|  |  |
| --- | --- |
| Figure 5.1 line 19 indicating the index type[[2]](#footnote-2) | Figure 5.2: MySQL support engines |

To support foreign key mechanism in our database, we need to use the InnoDB engine in our database construction since it is the only engine in MySQL providing this mechanism (which you can checked by typing “show engine” in MySQL).

Whereas InnoDB does not support creating hashing index manually, instead it has a mechanism called Adaptive Hash Index, which will set the index to be hash like basing on its observation. It can be set up by turning on global variable innodb\_adaptive\_hash\_index.

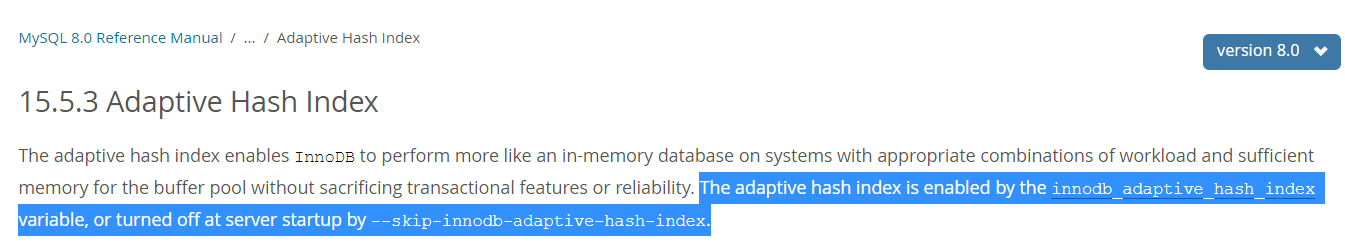


Figure 5.3: innodb adaptive hash index[[3]](#footnote-3)

Therefore, we need to turn set global variable innodb\_adaptive\_hash\_index = ON.

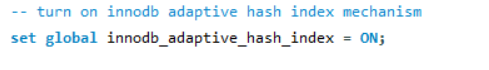


Figure 5.4: set innodb\_adaptive\_hash\_index = ON

Therefore we only need to set all the index to B-tree which is the default setting and InnoDB will change it to hash when necessary.

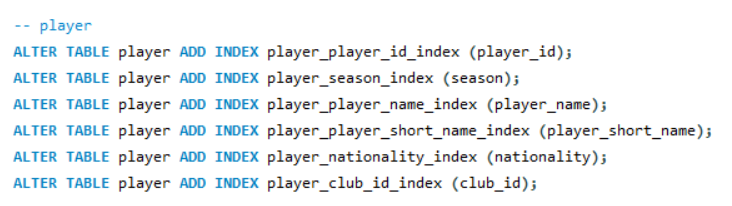


Figure 5.5: add indexes to table (partial)

After creating indexes, the performance of retrieving data using exact search and prefix fuzzy search will be improved a lot. The following is the test sample which use FLUSH operation and no-query-cache to avoid query cache affect the testing result. Notice that the query with index is deliberately be tested before the query after dropping index, which means if the memory still cache data, it will benefit the later query. With this setting, the later one without index still performs much worse than the previous one with index.

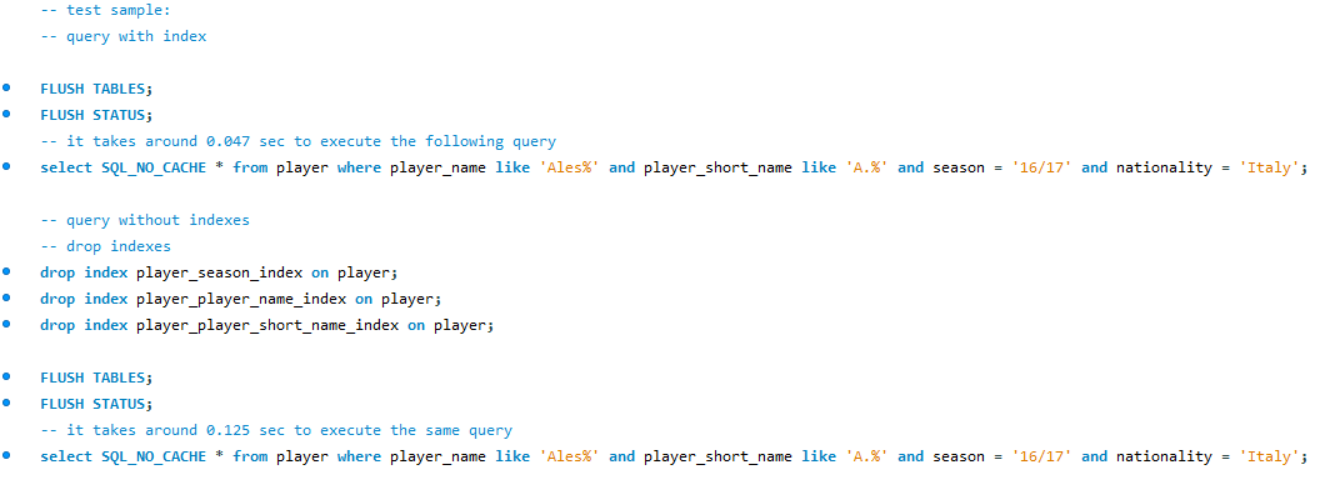


Figure 5.6: test sample for index

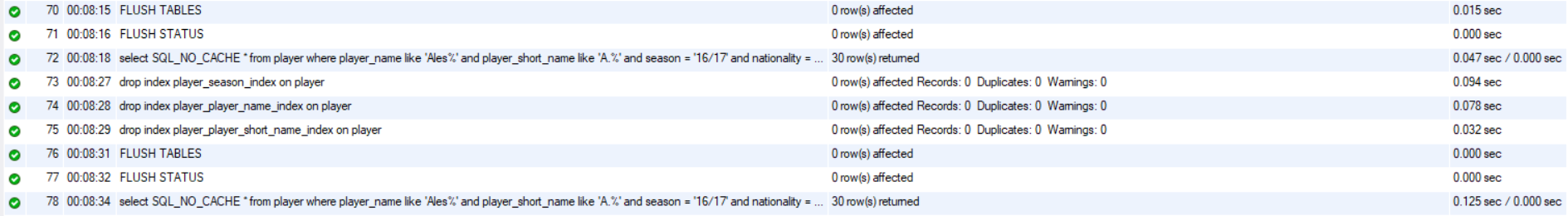


Figure 5.7: result of test sample

Notice in the above figure, query 73 (select with index) performs much better than query 78 (select without index). The sql codes in this section are written in /FIFADB/Codes/ step\_4\_create\_indexes.sql.

**Limitations & Further Improvements:**

1. This database does not contain statistics related to football matches. For example, the scorelines, lineups, numbers of goals and assists of players, numbers of yellow cards of players, numbers of saves of goalkeepers, leaderboards of leagues.

2. This database does not contain details of transfers and loans. For example, transfer fees.

3. This database does not contain information of national teams. For example, national team rosters.

4. This database does not contain detailed information about clubs. For example, trainings and staff.

The above information can be very useful in decision support. Our database can be further improved by incorporating this set of information.

**Appendix:**

|  |  |  |
| --- | --- | --- |
| club\_name | season | total wage (€) |
| Arsenal | 14/15 | 2428000 |
| Arsenal | 15/16 | 2640000 |
| Arsenal | 16/17 | 3146000 |
| Arsenal | 17/18 | 2961000 |
| Arsenal | 18/19 | 2484000 |
| Arsenal | 19/20 | 1996000 |
| Arsenal | 20/21 | 2046000 |
| Chelsea | 14/15 | 2571000 |
| Chelsea | 15/16 | 2847000 |
| Chelsea | 16/17 | 3379000 |
| Chelsea | 17/18 | 3508000 |
| Chelsea | 18/19 | 3367000 |
| Chelsea | 19/20 | 2775000 |
| Chelsea | 20/21 | 2698000 |
| Liverpool | 14/15 | 1832000 |
| Liverpool | 15/16 | 2111000 |
| Liverpool | 16/17 | 2514000 |
| Liverpool | 17/18 | 2725000 |
| Liverpool | 18/19 | 2996000 |
| Liverpool | 19/20 | 2626000 |
| Liverpool | 20/21 | 3154000 |
| Manchester City | 14/15 | 2830000 |
| Manchester City | 15/16 | 2817000 |
| Manchester City | 16/17 | 3442000 |
| Manchester City | 17/18 | 3141000 |
| Manchester City | 18/19 | 3776000 |
| Manchester City | 19/20 | 3958000 |
| Manchester City | 20/21 | 3765000 |
| Manchester United | 14/15 | 2659000 |
| Manchester United | 15/16 | 2278000 |
| Manchester United | 16/17 | 3412000 |
| Manchester United | 17/18 | 3497000 |
| Manchester United | 18/19 | 3483000 |
| Manchester United | 19/20 | 2837000 |
| Manchester United | 20/21 | 2950000 |
| Tottenham Hotspur | 14/15 | 1860000 |
| Tottenham Hotspur | 15/16 | 1577000 |
| Tottenham Hotspur | 16/17 | 2045000 |
| Tottenham Hotspur | 17/18 | 2240000 |
| Tottenham Hotspur | 18/19 | 2542000 |
| Tottenham Hotspur | 19/20 | 2563500 |
| Tottenham Hotspur | 20/21 | 2542000 |

**Table 1**

|  |  |  |  |
| --- | --- | --- | --- |
| league\_name | club\_name | total\_wage\_eur | rank |
| Spain Primera Division | Real Madrid | 4848000 | 1 |
| Spain Primera Division | FC Barcelona | 4738000 | 2 |
| Spain Primera Division | Atlético Madrid | 1597000 | 3 |
| Spain Primera Division | Sevilla FC | 814000 | 4 |
| Spain Primera Division | Villarreal CF | 813500 | 5 |
| Spain Primera Division | Valencia CF | 714900 | 6 |
| Spain Primera Division | Real Sociedad | 691000 | 7 |
| Spain Primera Division | CA Osasuna | 612500 | 8 |
| Spain Primera Division | Real Betis | 585000 | 9 |
| Spain Primera Division | Getafe CF | 565500 | 10 |
| Spain Primera Division | Levante UD | 563000 | 11 |
| Spain Primera Division | SD Eibar | 563000 | 11 |
| Spain Primera Division | Athletic Club de Bilbao | 563000 | 11 |
| Spain Primera Division | Granada CF | 539000 | 14 |
| Spain Primera Division | Deportivo Alavés | 495000 | 15 |
| Spain Primera Division | Cádiz CF | 492000 | 16 |
| Spain Primera Division | RC Celta | 447300 | 17 |
| Spain Primera Division | Real Valladolid CF | 435000 | 18 |
| Spain Primera Division | SD Huesca | 345000 | 19 |
| Spain Primera Division | Elche CF | 208950 | 20 |
| German 1. Bundesliga | FC Bayern München | 1802000 | 1 |
| German 1. Bundesliga | Borussia Dortmund | 1314000 | 2 |
| German 1. Bundesliga | Bayer 04 Leverkusen | 1230000 | 3 |
| German 1. Bundesliga | RB Leipzig | 1044550 | 4 |
| German 1. Bundesliga | VfL Wolfsburg | 888000 | 5 |
| German 1. Bundesliga | Borussia Mönchengladbach | 854600 | 6 |
| German 1. Bundesliga | Hertha BSC | 769500 | 7 |
| German 1. Bundesliga | TSG 1899 Hoffenheim | 725000 | 8 |
| German 1. Bundesliga | Eintracht Frankfurt | 585000 | 9 |
| German 1. Bundesliga | FC Schalke 04 | 505000 | 10 |
| German 1. Bundesliga | FC Augsburg | 504000 | 11 |
| German 1. Bundesliga | SV Werder Bremen | 502000 | 12 |
| German 1. Bundesliga | 1. FC Köln | 456000 | 13 |
| German 1. Bundesliga | SC Freiburg | 448000 | 14 |
| German 1. Bundesliga | 1. FSV Mainz 05 | 448000 | 14 |
| German 1. Bundesliga | 1. FC Union Berlin | 440600 | 16 |
| German 1. Bundesliga | VfB Stuttgart | 361300 | 17 |
| German 1. Bundesliga | DSC Arminia Bielefeld | 315000 | 18 |
| French Ligue 1 | Paris Saint-Germain | 2125550 | 1 |
| French Ligue 1 | AS Monaco | 1054000 | 2 |
| French Ligue 1 | Olympique Lyonnais | 1004000 | 3 |
| French Ligue 1 | Olympique de Marseille | 780950 | 4 |
| French Ligue 1 | Stade Rennais FC | 749000 | 5 |
| French Ligue 1 | LOSC Lille | 587150 | 6 |
| French Ligue 1 | OGC Nice | 546000 | 7 |
| French Ligue 1 | FC Girondins de Bordeaux | 534000 | 8 |
| French Ligue 1 | Racing Club de Lens | 513000 | 9 |
| French Ligue 1 | Montpellier HSC | 487000 | 10 |
| French Ligue 1 | AS Saint-Étienne | 433100 | 11 |
| French Ligue 1 | RC Strasbourg Alsace | 415000 | 12 |
| French Ligue 1 | FC Nantes | 404000 | 13 |
| French Ligue 1 | Angers SCO | 361000 | 14 |
| French Ligue 1 | Stade de Reims | 350000 | 15 |
| French Ligue 1 | FC Lorient | 320000 | 16 |
| French Ligue 1 | FC Metz | 291500 | 17 |
| French Ligue 1 | Nîmes Olympique | 286000 | 18 |
| French Ligue 1 | Stade Brestois 29 | 243000 | 19 |
| French Ligue 1 | Dijon FCO | 183100 | 20 |
| English Premier League | Manchester City | 3765000 | 1 |
| English Premier League | Liverpool | 3154000 | 2 |
| English Premier League | Manchester United | 2950000 | 3 |
| English Premier League | Chelsea | 2698000 | 4 |
| English Premier League | Tottenham Hotspur | 2542000 | 5 |
| English Premier League | Arsenal | 2046000 | 6 |
| English Premier League | Leicester City | 1981000 | 7 |
| English Premier League | Everton | 1872000 | 8 |
| English Premier League | Fulham | 1616000 | 9 |
| English Premier League | Wolverhampton Wanderers | 1344000 | 10 |
| English Premier League | Leeds United | 1212500 | 11 |
| English Premier League | West Bromwich Albion | 1161000 | 12 |
| English Premier League | Aston Villa | 1141000 | 13 |
| English Premier League | Crystal Palace | 1131000 | 14 |
| English Premier League | Brighton & Hove Albion | 1068000 | 15 |
| English Premier League | Southampton | 1021000 | 16 |
| English Premier League | Newcastle United | 993000 | 17 |
| English Premier League | Burnley | 926000 | 18 |
| English Premier League | West Ham United | 850000 | 19 |
| English Premier League | Sheffield United | 634000 | 20 |
| Italian Serie A | Inter | 2776000 | 1 |
| Italian Serie A | Juventus | 2161000 | 2 |
| Italian Serie A | Napoli | 1833000 | 3 |
| Italian Serie A | Lazio | 1513000 | 4 |
| Italian Serie A | Atalanta | 1297000 | 5 |
| Italian Serie A | Roma | 1092000 | 6 |
| Italian Serie A | Fiorentina | 977550 | 7 |
| Italian Serie A | Torino | 862000 | 8 |
| Italian Serie A | Milan | 807000 | 9 |
| Italian Serie A | Sassuolo | 589000 | 10 |
| Italian Serie A | Parma | 569000 | 11 |
| Italian Serie A | Bologna | 548000 | 12 |
| Italian Serie A | Cagliari | 536000 | 13 |
| Italian Serie A | Hellas Verona | 466000 | 14 |
| Italian Serie A | Spezia | 360000 | 15 |
| Italian Serie A | Udinese | 345450 | 16 |
| Italian Serie A | Genoa | 325700 | 17 |
| Italian Serie A | Sampdoria | 310000 | 18 |
| Italian Serie A | Benevento | 285900 | 19 |
| Italian Serie A | Crotone | 150300 | 20 |

**Table 2.1**

|  |  |
| --- | --- |
| league\_name | average |
| French Ligue 1 | 583367.5 |
| German 1. Bundesliga | 732919.4 |
| Italian Serie A | 890195 |
| Spain Primera Division | 1031533 |
| English Premier League | 1705275 |

**Table 2.2**

|  |  |
| --- | --- |
| league\_name | standard deviation |
| German 1. Bundesliga | 385994.7 |
| French Ligue 1 | 422336.4 |
| Italian Serie A | 686719.2 |
| English Premier League | 865735.8 |
| Spain Primera Division | 1281632 |

**Table 2.3**

|  |  |  |  |
| --- | --- | --- | --- |
| club\_name | season | position\_class | average rating |
| Atlético Madrid | 18/19 | defender | 75.1429 |
| Atlético Madrid | 18/19 | forward | 74.4545 |
| Atlético Madrid | 18/19 | midfielder | 75.7143 |
| Atlético Madrid | 18/19 | NULL | 75.0313 |
| Atlético Madrid | 19/20 | defender | 76.3571 |
| Atlético Madrid | 19/20 | forward | 73.1538 |
| Atlético Madrid | 19/20 | midfielder | 77.3 |
| Atlético Madrid | 19/20 | NULL | 75.4865 |
| Atlético Madrid | 20/21 | defender | 77.1429 |
| Atlético Madrid | 20/21 | forward | 74.1429 |
| Atlético Madrid | 20/21 | midfielder | 77.5833 |
| Atlético Madrid | 20/21 | NULL | 76.225 |
| Atlético Madrid | NULL | NULL | 75.6239 |
| FC Barcelona | 18/19 | defender | 77.7857 |
| FC Barcelona | 18/19 | forward | 79.75 |
| FC Barcelona | 18/19 | midfielder | 80.6667 |
| FC Barcelona | 18/19 | NULL | 79.2647 |
| FC Barcelona | 19/20 | defender | 76.7647 |
| FC Barcelona | 19/20 | forward | 77 |
| FC Barcelona | 19/20 | midfielder | 78.8333 |
| FC Barcelona | 19/20 | NULL | 77.5143 |
| FC Barcelona | 20/21 | defender | 78.6923 |
| FC Barcelona | 20/21 | forward | 77.7273 |
| FC Barcelona | 20/21 | midfielder | 79.0714 |
| FC Barcelona | 20/21 | NULL | 78.5526 |
| FC Barcelona | NULL | NULL | 78.4393 |
| Real Madrid | 18/19 | defender | 78.0588 |
| Real Madrid | 18/19 | forward | 76.7778 |
| Real Madrid | 18/19 | midfielder | 80.1818 |
| Real Madrid | 18/19 | NULL | 78.3784 |
| Real Madrid | 19/20 | defender | 78 |
| Real Madrid | 19/20 | forward | 78.0909 |
| Real Madrid | 19/20 | midfielder | 79.5 |
| Real Madrid | 19/20 | NULL | 78.5641 |
| Real Madrid | 20/21 | defender | 78.3846 |
| Real Madrid | 20/21 | forward | 77.6667 |
| Real Madrid | 20/21 | midfielder | 80.3333 |
| Real Madrid | 20/21 | NULL | 78.7838 |
| Real Madrid | NULL | NULL | 78.5752 |

**Table 3**

**Figure A1**

**Figure A1**

**Figure A2**

**Figure A3**

**Figure A4**

**Figure A5**

**Figure A6**

**Figure A7**

**Figure A8**

**Figure A9**

**Figure A10**

**Query Code 1-5**

1.Search player data:

Results=Website.query.filter(

            Website.season.like('%'+season+'%') if season is not None else text(""),

            Website.club\_id.in\_(Club\_id) if league\_id is not None else text(""),

            Website.nationality.like('%'+nationality+'%') if nationality is not None else text(""),

            Website.player\_name.like('%'+player\_name+'%') if player\_name is not None else text(""),

            Website.club\_name.like('%'+club\_name+'%') if player\_name is not None else text(""),

            Website.overall.like('%'+overall+'%') if overall is not None else text(""),

            Website.potential.like('%'+potential+'%') if potential is not None else text(""),

        ).all()

2.Search league data:

results=db.session.query(League.league\_id, League.season, League.league\_name, League.country\_region\_name, Club.club\_name).filter(League.league\_id.like(league\_id) if league\_id is not None else text(""),

        League.season.like('%'+season+'%') if season is not None else text(""),

        Club.season.like('%'+season+'%') if season is not None else text(""),

        Club.league\_id.like('%'+league\_id+'%') if season is not None else text(""),

        Club.club\_name.like('%'+club\_name+'%') if season is not None else text("")).all()

3.Insert player:

new\_player = New\_Player(player\_id=player\_id,season=season,player\_name=player\_name,nationality=nationality,club\_id=club\_id,overall=overall,potential=potential)

                print(new\_player)

                db.session.add(new\_player)

                db.session.commit()

4.Delete player:

q\_number=Website.query.filter\_by(player\_id=player\_id).first()

if(delete):

             db.session.delete(q\_number)

5.Edit player:

q\_number=Website.query.filter\_by(player\_id=player\_id).first()

if(nationality!=q\_number.nationality):

                   q\_number.nationality = nationality

                   db.session.commit()

                if(player\_name!=q\_number.player\_name):

                    q\_number.player\_name = player\_name

                    db.session.commit()

                if(club\_id!=q\_number.club\_id):

                    q\_number.club\_id = club\_id

                    db.session.commit()

                if(overall!=q\_number.overall):

                    q\_number.overall = overall

                    db.session.commit()

                if(potential!=q\_number.potential):

                    q\_number.potential = potential

                    db.session.commit()

1. Dataset is retrieved from https://www.kaggle.com/bryanb/fifa-player-stats-database. [↑](#footnote-ref-1)
2. Retrieved from https://dev.mysql.com/doc/refman/8.0/en/create-index.html [↑](#footnote-ref-2)
3. https://dev.mysql.com/doc/refman/8.0/en/innodb-adaptive-hash.html [↑](#footnote-ref-3)