

databases final project

FIFA18 Video Game Database



December 16, 2017

Ye Chan Kim (315) | Jae Goan park (315)

ykim160@jhu.edu | jpark230@jhu.edu

Our database consists of various elements from the popular video game FIFA 18. The database includes soccer players’ personal information, their attributes, quantifiable skill level and associated clubs.

[ugrad.cs.jhu.edu/~ykim160/fifa.php](http://ugrad.cs.jhu.edu/~ykim160/fifa.php)

**I. Change from Phase I**

We have made some design changes to the FIFA 18 database. We decided to remove the continent relation as it did not directly relate to the goals of the project and instead added more parts to make it easier for a user to navigate and make queries to the database. In addition to the Offensive(Attack) and Defensive(Defense) skill relations we’ve also added the midfield skill relation to cover a broader range of players who may fall equidistantly between the three different categories.

Also we decided to add a new relation called “Country” that contains all the country FIFA codes as Key with the associated country names. This meant that we removed the country\_name column from the Player relation and instead added “NTC” which contains all the 3 letter FIFA country code abbreviations. Also in the Country relation, we have the country’s capital, Continent Abbreviation, sub-region, and its development status (whether or not it is a developed country).

We also made some changes to the soccer team relation. We removed the player’s associated football club from the Player relation, and instead created another relation called “Plays\_in” to link with the Player\_id from the player relation to the “Club” relation. The Club relation contains all the “Club\_id” numbers with the associated club name.

Finally, we’ve also added a attributes relation that contains a miscellaneous set of quantifiable attribute statistics to cover a broader range of skills for individual players.

**II. Loading the Database with values**

We obtained our dataset to populate out database with from Kaggle (<https://www.kaggle.com/thec03u5/fifa-18-demo-player-dataset)>, which is a renowned data science website that publishes many datasets for a variety of applications. The particular dataset contained all the data we needed to populate our data base in csv format. We used Pandas, which is a python library, to extract data from the different csv files contained in the dataset. To make the debugging process easier, we made a data extraction tool through python in several different files. Simply, each file that ends with “\_extraction.py” is responsible for extracting the data for each relation. The method of extraction was fairly simple due to the help of the pandas library. We used the pandas.read\_csv(filename, header etc.) command to read in the csv files and physically print the necessary sql commands to a sql file. We wrote to files the commands such as “DROP IF EXISTS Player” and “INSERT INTO Player” and filled out the necessary data parts by using a for loop to loop over the data extracted using the read\_csv method. The individual files that contained the extraction code outputted a sql file as a result and we ran the individual sql files on MySQL and was able to successfully populate the database.

**III. User’s guide**

1. Before proceeding with running the codes, please be sure to have either python2.7 or python3 installed each equipped with the pandas and numpy library written for the python programming language.
   1. To install pandas for python2.7: pip install pandas
   2. To Install pandas for python3: pip3 install pandas
2. Now to create the necessary sql files needed to populate the database we must run each individual \*\_extraction.py file on the command line. To run the individual python codes, please use the command: python3 replace\_with\_your\_file.sql
   1. Please first run the file createSQL.py with command python createSQL.py to create another csv file necessary for country extraction with proper FIFA abbreviation mapping
   2. Python file list: Attribute\_extraction.py, club\_extraction.py, country\_extraction.py, player\_extraction.sql, player\_preferred\_extraction.sql, player\_Value\_Extraction.py, skills\_attack\_extraction.py, skills\_defense\_extraction.py, skills\_midfield\_extraction.py
3. Once all the necessary sql files have been created, log into dbase and run the following command: \. ~/path\_to\_directory\_containing\_sql\_files/sql\_file
   1. Sql file list: attack.sql, attribute.sql, club.sql, country.sql, defense.sql, midfield.sql, player.sql, plays\_in.sql, Position.sql, Preferred.sql, Value.sql
4. As we are running the codes in dbase, please place all the php and html files in the zip file inside the public\_html folder in the home directory of your ugrad virtual machine.
   1. Move into the public\_html director
   2. Set permissions for all the php files with the command: chmod 700 \*.php and all the html files with the command: chmod 604 \*.html inside the public\_html directory.

**IV. Major/Minor areas of specialization**

Initially we focused on complex data extraction issues, as we thought we would be facing severe issues with the different encodings and character sets of the different player names and club names obtained from a worldwide database. Towards the beginning of the project, we were unable to fix the issues with broken up alphabets coming from non-standard ASCII characters. Some foreign letters such as “ø” or “ü” appeared as question marks, and this was a database wide issue as we had successful extraction, but had continuous issues with displaying the original characters used to describe player names or club names. We researched this issue to try and pre-process the dataset to encode all the strings into UTF-8 format before populating the database, and have also tried normalizing the names into standard English alphabet characters, but realized that this could take too long. Towards the end of the project, we realized that the issue could be solved rather easily by changing what the database can accept itself. Dbase on JHU’s server was for some reason set at default to latin1 encoding, which was insufficient to fully list our database in proper condition. The one-liner that resolved the encoding issue turned out to be:

1. **ALTER** **DATABASE** YOUR\_DBNAME **CHARACTER** **SET** utf8 **COLLATE** utf8\_general\_ci;

But we still had a lingering issue in terms of suiting the needs of our predesigned relational models. The original dataset we had obtained had some issues as it did not contain the appropriate country codes for country names, and therefore we did not have a primary key for the Country relation. To resolve this issue, we had to obtain another dataset from another source (<https://github.com/openmundi/world.csv)> and process this data accordingly. We were also faced with the problem where the country names obtained from the new data set did not exactly match the original country names we had(e.g. South Korea written as Republic of Korea etc). To approach this issue, we extracted country names from PlayerPersonalData.csv and the newly obtained country-codes.csv to see which data from both datasets actually matched by running a 1to1 comparison. For those with matches, we created as dictionary to include the country names as keys and the abbreviations as the values matching the keys. This increased work productivity as we only needed to hard-code in outliers to perfect the dataset as a whole.

**V. Strengths and selling points**

We believe that we have created a user front-end interface that is visually pleasing and fairly easy to use. We’ve divided the user interface into 4 sectors on a single webpage so that the user can navigate between the different functions of the 4 different pages through a single link.

We also believe that we have done a good amount of basework to alleviate encoding issues with the imported names and soccer club names. As many of the proper nouns obtained in the dataset had some components that were not part of the standard ASCII character set, we struggled at first to obtain a fully comprehensive database due to some characters being broken up in the process. However, we were able to fix the issues with writing code inside the sql files that decoded and set the correct parameters for foreign characters and by changing the defauly settings of acceptable language on dbase to utf8 from latin1. We realized that we were able to simplify this process by changing the acceptable standards of the database rather than process the dataset itself to fit into the database.

**VI. Limitations and suggested improvements**

We believe that we could have done a better job with normalization of the names of the soccer players. As in the real game FIFA18, some of the players’ names were not given in the standard format where the player’s first initial is followed by his full last name(e.g. C. Ronaldo). Some names only contained last names and therefore caused some issues with being able to sort the names in easy-to-search proper order. In order to improve this issue, we believe that it would be better to use data mining applications and instead obtain data from the official soccer federation website to obtain last and first names separately to resolve the sorting issue.

We also hope with that more time, we could actually conduct surveys to decide which factors are most important in building an optimal team for FIFA18 video gamers. It was slightly difficult to decide which factors to include when asking the user to input preferences as there were way more factors that mattered than we had previously expected and therefore it would not make sense to ask the user a handful of questions to decide which team would be the best. Instead we just created an all-in-one interface where the user can input country, team preferences to find players and look up the skills of the individual players to decide whether or not these players are a match for his/her team.

**VII. Code from elsewhere**

Pandas Library (Python)

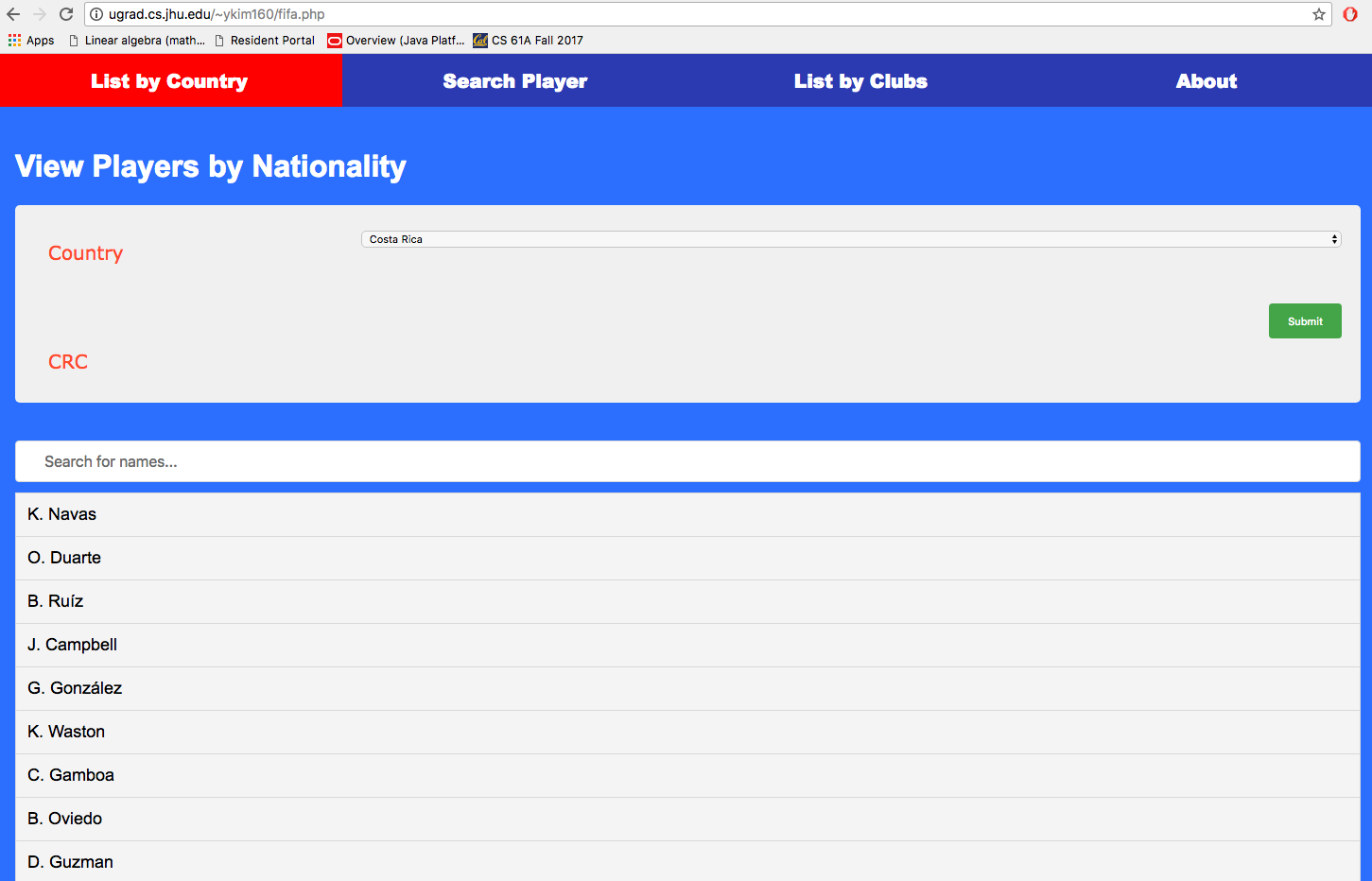
To extract data from the given csv files, we imported the python pandas library to assist the process.

W3schools (php and html template)

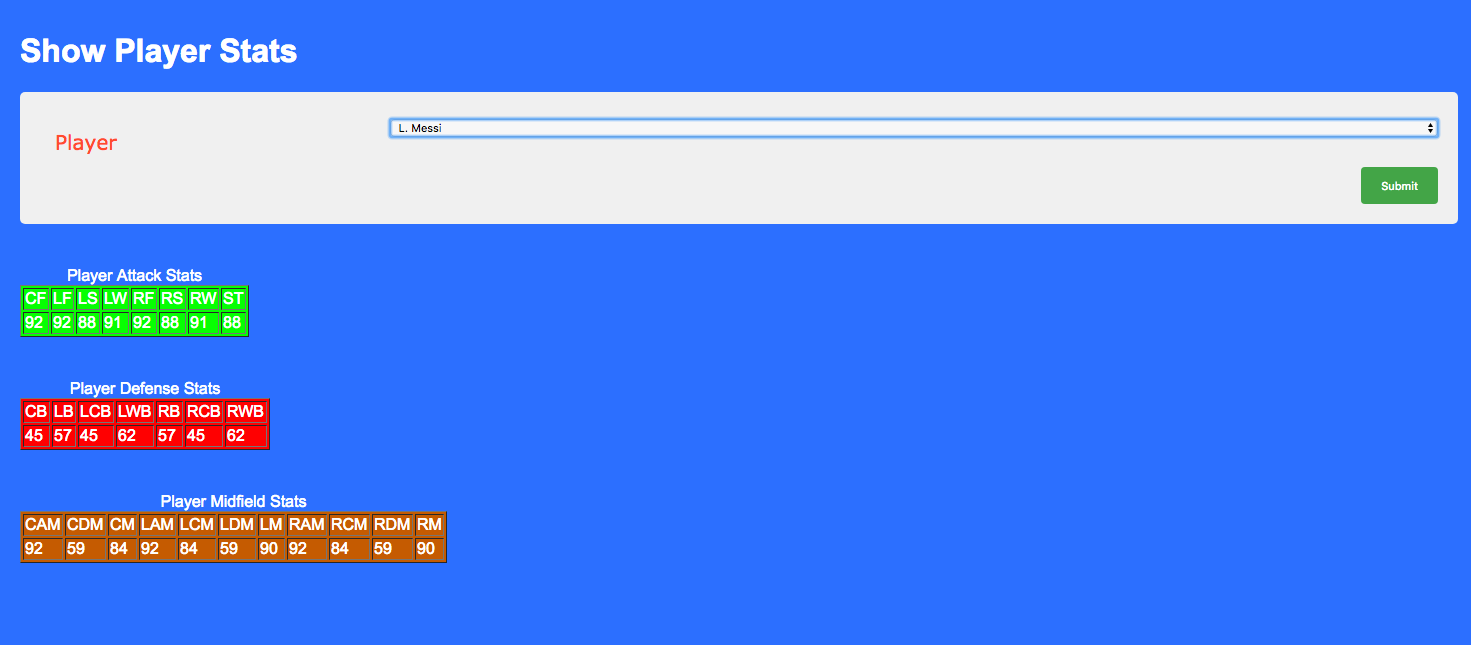
Also while we have mostly self learned the php components of creating a front-end interface for the user, to make the design more visually appealing, we integrated our php code into templates we were able to find from a tutorial site called w3schools. While learning how to create drop down menus, we found it easier to embed contents of the html file into the php file, so we decided to combine the two files instead.

**VIII. Project Output**

1. List a specific player’s country of origin, team, and age
2. mysql> **SELECT** P.**Name**, P.Age, T.Club\_name, C.Country\_name **FROM** Player **AS** P, Plays\_in **as** I, Club **as** T, Country **AS** C **WHERE** P.Player\_id=I.Player\_id AND I.Club\_id=T.Club\_id AND P.**Name**="Cristiano Ronaldo" AND C.NTC=P.NTC;
3. +-------------------+------+----------------+--------------+
4. | **Name**              | Age  | Club\_name      | Country\_name |
5. +-------------------+------+----------------+--------------+
6. | Cristiano Ronaldo |   32 | **Real** Madrid CF | Portugal     |
7. +-------------------+------+----------------+--------------+
8. List all players and their respective football clubs who have a “striker” skill over 90
9. mysql> **SELECT** P.**Name**, C.Club\_name, P.Age
10. -> **FROM** (**SELECT** Player\_id **FROM** Attack **WHERE** ST>=85) **AS** Z,
11. -> Player **AS** P, Plays\_in **AS** I, Club **AS** C
12. -> **WHERE** Z.Player\_id=P.Player\_id AND P.Player\_id=I.Player\_id AND I.CLub\_id=C.Club\_id;
13. +-------------------+-------------------+------+
14. | **Name**              | Club\_name         | Age  |
15. +-------------------+-------------------+------+
16. | Cristiano Ronaldo | **Real** Madrid CF    |   32 |
17. | L. Messi          | FC Barcelona      |   30 |
18. | L. Suárez         | FC Barcelona      |   30 |
19. | R. Lewandowski    | FC Bayern Munich  |   28 |
20. | G. Higuaín        | Juventus          |   29 |
21. | G. Bale           | **Real** Madrid CF    |   27 |
22. | S. Agüero         | Manchester City   |   29 |
23. | A. Griezmann      | Atlético Madrid   |   26 |
24. | P. Aubameyang     | Borussia Dortmund |   28 |
25. +-------------------+-------------------+------+
26. 9 **rows** in **set** (0.70 sec)
27. Average age of players from real madrid
28. mysql> **SELECT** Z.Club\_name, AVG(Z.Age) **AS** Avg\_Age
29. -> **FROM** (**SELECT** C.Club\_name, P.Age
30. -> **FROM** Player **AS** P, Plays\_in **AS** I, Club **AS** C
31. -> **WHERE** P.Player\_id=I.Player\_id AND I.Club\_id=C.CLub\_id AND C.Club\_name="Real Madrid CF") **AS** Z;
32. +----------------+---------+
33. | Club\_name      | Avg\_Age |
34. +----------------+---------+
35. | **Real** Madrid CF | 24.3750 |
36. +----------------+---------+
37. 1 row in **set** (0.05 sec)
38. List the name and defense skill levels of the best Left\_Back player
39. mysql> **SELECT** P.**Name**, Z.CB, Z.LB, Z.LCB, Z.LWB, Z.RB
40. -> **FROM** (**SELECT** \*
41. -> **FROM** Defense
42. -> **ORDER** **BY** LB **DESC**
43. -> LIMIT 1) **AS** Z, Player **AS** P
44. -> **WHERE** P.Player\_id=Z.Player\_id;
45. +--------------+------+------+------+------+------+
46. | **Name**         | CB   | LB   | LCB  | LWB  | RB   |
47. +--------------+------+------+------+------+------+
48. | Sergio Ramos |   87 |   84 |   87 |   81 |   84 |
49. +--------------+------+------+------+------+------+
50. 1 row in **set** (0.02 sec)
51. **Webpage output that enables search of players by country (Costa Rica)**

****

1. **List Player stats by player name**

****

**VIIII. Relational Table Specification**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Player** | Player\_id | Name | Age | Overall | NTC |
|  | 184941 | A. Sánchez | 28 | 89 | CHI |

|  |  |  |
| --- | --- | --- |
| **Plays\_in** | Player\_ID | Club\_id |
|  | 182521 | 349 |

|  |  |  |
| --- | --- | --- |
| **club** | Club\_id | Club\_Name |
|  | 14 | SC Paderborn 07 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Country** | NTC | Country\_name | Capital | Development | sub\_region |
|  | AUS | Austrailia | Canberra | Developed | Australia and New Zealand |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Attack** | Player\_id | CF | LF | LS | LW | RF | RS | RW | ST |
|  | 20801 | 91 | 91 | 92 | 91 | 91 | 92 | 91 | 92 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Midfield** | Player\_id | CAM | CDM | CM | LAM | LCM | LDM |
|  | 182521 | 83 | 82 | 87 | 83 | 82 | 81 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Defense** | Player\_id | CB | LB | LCB | LWB | RB | RCB | RWB |
|  | 188545 | 57 | 58 | 57 | 61 | 58 | 57 | 61 |

|  |  |  |
| --- | --- | --- |
| **Positions** | Position\_Abv | Position\_Name |
|  | CAM | Center\_Attacking\_Midfield |

|  |  |  |
| --- | --- | --- |
| **Preferred** | Player\_id | Preferred\_Position |
|  | 188545 | ST |

|  |  |  |  |
| --- | --- | --- | --- |
| **Value** | Player\_id | Player\_Value | Player\_Wage |
|  | 153079 | €66.5M | €325K |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Attributes** | Player\_id | Acceleration | Aggression | Agility | Balance | Ball\_control | Composure | Crossing | Curve |
|  | 138956 | 68 | 92 | 59 | 64 | 57 | 82 | 58 | 60 |

**X. Code Hard Copy**

**Python code used to extract data from csv format**

(Attribute\_extraction.py)

1. #Please run this code with the command python3 Attribute\_extraction.py
3. #import pandas library
4. **import** pandas as pd
6. #read in PlayerAttributeData.csv file into pandas dataframe
7. attribute\_data = pd.read\_csv('PlayerAttributeData.csv', low\_memory=False)
9. #code to test if csv file was read in
10. #print(attribute\_data)
12. #get the total number of rows and cols
13. total\_rows = attribute\_data.shape[0]
14. total\_cols = attribute\_data.shape[1]
15. #create new sql file, attribute.sql and write to it the commands necessary to populate data this sql file will be run later
16. f = open("attribute.sql", "w")
17. **print**("DROP TABLE IF EXISTS Attributes; \nCREATE TABLE Attributes (ID INT, Acceleration INT, Aggression INT, Agility INT, Balance INT, Ball\_control INT, Composure INT, Crossing INT, Curve INT, Dribbling INT, Finishing INT, Free\_kick\_accuracy INT, GK\_diving INT, GK\_handling INT, GK\_kicking INT, GK\_positioning INT, GK\_reflexes INT, Heading\_accuracy INT, Interceptions INT, Jumping INT, Long\_passing INT, Long\_shots INT, Marking INT, Penalties INT, Positioning INT, Reactions INT, Short\_passing INT, Shot\_power INT, Sliding\_tackle INT, Sprint\_speed INT, Stamina INT, Standing\_tackle INT, Strength INT, Vision INT, Volleys INT);\n", file=f)
19. **for** i **in** range(0,total\_rows):
20. **print**("INSERT into Attributes values (", attribute\_data["ID"][i],
21. ", ", eval(str(attribute\_data["Acceleration"][i])),
22. ", ", eval(str(attribute\_data["Aggression"][i])),
23. ", ", eval(str(attribute\_data["Agility"][i])),
24. ", ", eval(str(attribute\_data["Balance"][i])),
25. ", ", eval(str(attribute\_data["Ball control"][i])),
26. ", ", eval(str(attribute\_data["Composure"][i])),
27. ", ", eval(str(attribute\_data["Crossing"][i])),
28. ", ", eval(str(attribute\_data["Curve"][i])),
29. ", ", eval(str(attribute\_data["Dribbling"][i])),
30. ", ", eval(str(attribute\_data["Finishing"][i])),
31. ", ", eval(str(attribute\_data["Free kick accuracy"][i])),
32. ", ", eval(str(attribute\_data["GK diving"][i])),
33. ", ", eval(str(attribute\_data["GK handling"][i])),
34. ", ", eval(str(attribute\_data["GK kicking"][i])),
35. ", ", eval(str(attribute\_data["GK positioning"][i])),
36. ", ", eval(str(attribute\_data["GK reflexes"][i])),
37. ", ", eval(str(attribute\_data["Heading accuracy"][i])),
38. ", ", eval(str(attribute\_data["Interceptions"][i])),
39. ", ", eval(str(attribute\_data["Jumping"][i])),
40. ", ", eval(str(attribute\_data["Long passing"][i])),
41. ", ", eval(str(attribute\_data["Long shots"][i])),
42. ", ", eval(str(attribute\_data["Marking"][i])),
43. ", ", eval(str(attribute\_data["Penalties"][i])),
44. ", ", eval(str(attribute\_data["Positioning"][i])),
45. ", ", eval(str(attribute\_data["Reactions"][i])),
46. ", ", eval(str(attribute\_data["Short passing"][i])),
47. ", ", eval(str(attribute\_data["Shot power"][i])),
48. ", ", eval(str(attribute\_data["Sliding tackle"][i])),
49. ", ", eval(str(attribute\_data["Sprint speed"][i])),
50. ", ", eval(str(attribute\_data["Stamina"][i])),
51. ", ", eval(str(attribute\_data["Standing tackle"][i])),
52. ", ", eval(str(attribute\_data["Strength"][i])),
53. ", ", eval(str(attribute\_data["Vision"][i])),
54. ", ", eval(str(attribute\_data["Volleys"][i])),");",sep="", file=f)
56. f.close()

(club\_extraction.py)

1. # run with python2.7 with command python club\_extraction.py
2. **import** pandas as pd
4. personal\_data = pd.read\_csv('PlayerPersonalData.csv', keep\_default\_na=False)
6. personal\_total\_rows = personal\_data.shape[0]
8. club = []
9. id\_dict = {}
10. **for** i **in** range(personal\_total\_rows):
11. club.append(personal\_data["Club"][i])
12. id\_dict[personal\_data["ID"][i]] = personal\_data["Club"][i]
14. myset = set(club)
15. club = list(myset)
17. # plays\_in\_file = open('plays\_in.sql', 'w')
18. #
19. # plays\_in\_file.write("DROP TABLE IF EXISTS Plays\_in; \nCREATE TABLE Plays\_in (Player\_id INT, Club\_id INT);\n\n")
20. #
21. # for i in range(personal\_total\_rows):
22. #     index = 0
23. #     for j in range(len(club)):
24. #         if id\_dict[personal\_data["ID"][i]] == club[j]:
25. #             index = j
26. #     plays\_in\_file.write("INSERT into Plays\_in values (" + str(personal\_data["ID"][i]) + ", " + str(index) + ");\n")
27. #
28. # plays\_in\_file.close()
30. club\_file = open('club.sql', 'w')
32. club\_file.write("DROP TABLE IF EXISTS Club; \nCREATE TABLE Club (Club\_id INT, Club\_name VARCHAR(40));\n\n")
33. **for** i **in** range(len(club)):
34. club\_file.write("INSERT into Club values (" + str(i) + ", " + '"''"' + str(club[i]) + '"' + ");\n")
36. club\_file.close()

(country\_extraction.py)

1. **import** pandas as pd
2. **import** collections
4. country\_data = pd.read\_csv('country.csv', keep\_default\_na=False)
6. country\_total\_rows = country\_data.shape[0]
7. country\_file = open('country.sql', 'w')
9. country\_file.write("DROP TABLE IF EXISTS Country; \nCREATE TABLE Country (NTC VARCHAR(3), Country\_Name VARCHAR(20), "
10. "Capital VARCHAR(20), Continent VARCHAR(2), Development VARCHAR(20), Sub\_region VARCHAR(35));\n\n")
12. **for** i **in** range(country\_total\_rows):
13. country\_file.write("INSERT into Country values (" + '"''"' + str(country\_data["FIFA"][i]) + '"' + ", "
14. + '"''"' + str(country\_data["official\_name\_en"][i]) + '"' + ", "
15. + '"''"' + str(country\_data["Capital"][i]) + '"' + ", "
16. + '"''"' + str(country\_data["Continent"][i]) + '"' + ", "
17. + '"''"' + str(country\_data["Developed / Developing Countries"][i]) + '"' + ", "
18. + '"''"' + str(country\_data["Sub-region Name"][i]) + '"' + ");\n")
20. country\_file.close()

(createSQL.py)

1. **import** pandas as pd
2. **import** collections
3. **import** csv
5. personal\_data = pd.read\_csv('PlayerPersonalData.csv')
6. attribute\_data = pd.read\_csv('PlayerAttributeData.csv')
7. position\_data = pd.read\_csv('PlayerPlayingPositionData.csv')
8. country\_data = pd.read\_csv('og-country-codes.csv', keep\_default\_na=False)
10. personal\_total\_rows = personal\_data.shape[0]
11. # player\_file = open('player.sql', 'w')
12. #
13. # player\_file.write("DROP TABLE IF EXISTS Player; \nCREATE TABLE Player (Player\_id INT, Name VARCHAR(20), "
14. #                   "Nationality VARCHAR(20), Age INT, Team VARCHAR(20), Overall INT);\n\n")
15. #
16. # for i in range(personal\_total\_rows):
17. #     player\_file.write("INSERT into Player values (" + str(personal\_data["ID"][i]) + ", " + '"'
18. #                       + str(personal\_data["Name"][i]) + '"' + ", " + '"' + str(personal\_data["Nationality"][i]) + '"'
19. #                       + ", " + str(personal\_data["Age"][i]) + ", " + '"' + str(personal\_data["Club"][i]) + '"' + ", "
20. #                       + str(personal\_data["Overall"][i]) + ");\n")
21. #
22. # player\_file.close()
24. country = []
25. fifa = []
26. capital = []
27. continent = []
28. developed = []
29. region = []
30. country\_dic = {}
31. row = country\_data.shape[0]
32. **for** i **in** range(1,row):
33. country.append(country\_data["official\_name\_en"][i])
34. country\_dic[country\_data["official\_name\_en"][i]] = country\_data["FIFA"][i]
35. fifa.append(country\_data["FIFA"][i])
36. capital.append(country\_data["Capital"][i])
37. continent.append(country\_data["Continent"][i])
38. developed.append(country\_data["Developed / Developing Countries"][i])
39. region.append(country\_data["Sub-region Name"][i])
41. nationality = []
42. row2 = personal\_data.shape[0]
43. **for** i **in** range(row2):
44. nationality.append(personal\_data["Nationality"][i])

47. myset = set(nationality)
48. nationality = list(myset)
50. matched\_country = []
51. matched\_nationality = []
52. counter = 0
53. abv = {}
54. **for** i **in** range(len(nationality)):
55. **for** j **in** range(len(country)):
56. **if** nationality[i] == country[j]:
57. matched\_country.append(country[j])
58. matched\_nationality.append(nationality[i])
59. abv[nationality[i]] = country\_dic[country[j]]
61. not\_nationality = []
62. **for** i **in** range(len(nationality)):
63. **if** nationality[i] **not** **in** matched\_nationality:
64. not\_nationality.append(nationality[i])
66. abv["Palestine"] = "PLE"
67. abv["Bolivia"] = "BOL"
68. abv["Republic of Ireland"] = "IRL"
69. abv["Korea DPR"] = "PRK"
70. abv["FYR Macedonia"] = "MKD"
71. abv["DR Congo"] = "COD"
72. abv["Tanzania"] = "TAN"
73. abv["St Kitts Nevis"] = "SKN"
74. abv["England"] = "ENG"
75. abv["Kosovo"] = "RKS"
76. abv["Czech Republic"] = "CZE"
77. abv["Korea Republic"] = "KOR"
78. abv["United States"] = "USA"
79. abv["Russia"] = "RUS"
80. abv["Antigua & Barbuda"] = "ATG"
81. abv["Vietnam"] = "VIE"      # Not in the country-codes excel file
82. abv["Scotland"] = "SCT"
83. abv["Cape Verde"] = "CPV"
84. abv["Ivory Coast"] = "CIV"
85. abv["Wales"] = "WAL" # not in the country-codes excel file
86. abv["Iran"] = "IRN"
87. abv["Hong Kong"] = "HKG"
88. abv["China PR"] = "CHN"
89. abv["Guinea Bissau"] = "GNB"
90. abv["Moldova"] = "MDA"
91. abv[not\_nationality[25]] = "STP"
92. abv["Curacao"] = "CUW"
93. abv["Bosnia Herzegovina"] = "BIH"
94. abv["Venezuela"] = "VEN"
95. abv["Central African Rep."] = "CTA"
96. abv["Trinidad & Tobago"] = "TRI"
97. abv["St Lucia"] = "LCA"
98. abv["Syria"] = "SYR"
99. abv["Northern Ireland"] = "NIR"
101. alph\_abv = collections.OrderedDict()
102. **for** key, value **in** sorted(abv.items()):
103. alph\_abv[key] = value
105. # for i in range(len(country)):
106. #     if country[i] not in matched\_country:
107. #         print country[i]
109. country\_file = open('country.csv', 'wb')
110. writer = csv.writer(country\_file)
112. # index = fifa.index("AFG")
113. # print index
114. # print country[index], capital[index], continent[index], developed[index], region[index]
116. # print capital[233]
118. writer.writerow(["official\_name\_en"] + ["Capital"] + ["Continent"] + ["Developed / Developing Countries"]
119. + ["FIFA"] + ["Sub-region Name"])
120. **for** key, value **in** alph\_abv.items():
121. **if** value == "WAL":
122. writer.writerow([key] + ["Cardiff"] + ["EU"] + ["Developed"] + [value] + ["Northern Europe"])
123. **elif** value == "SCT":
124. writer.writerow([key] + ["Edinburgh"] + ["EU"] + ["Developed"] + [value] + ["Northern Europe"])
125. **elif** value == "NIR":
126. writer.writerow([key] + ["Belfast"] + ["EU"] + ["Developed"] + [value] + ["Northern Europe"])
127. **elif** value == "RKS":
128. writer.writerow([key] + ["Pristina"] + ["EU"] + ["Developing"] + [value] + ["Southern Europe"])
129. **elif** value == "CUW":
130. index = 57
131. writer.writerow([key] + [capital[index]] + [continent[index]] + [developed[index]] + [value] + [region[index]])
132. **elif** value == "ENG":
133. index = 233
134. writer.writerow([key] + [capital[index]] + [continent[index]] + [developed[index]] + [value] + [region[index]])
135. **else**:
136. index = fifa.index(value)
137. writer.writerow([key] + [capital[index]] + [continent[index]] + [developed[index]] + [value] + [region[index]])

(player\_extraction.py)

1. **import** pandas as pd
2. **import** collections
4. personal\_data = pd.read\_csv('PlayerPersonalData.csv')
5. attribute\_data = pd.read\_csv('PlayerAttributeData.csv')
6. position\_data = pd.read\_csv('PlayerPlayingPositionData.csv')
7. country\_data = pd.read\_csv('country.csv')
9. country\_dic = collections.OrderedDict()
10. country\_row = country\_data.shape[0]
11. **for** i **in** range(country\_row):
12. country\_dic[country\_data["official\_name\_en"][i]] = country\_data["FIFA"][i]
14. personal\_total\_rows = personal\_data.shape[0]
15. player\_file = open('player.sql', 'w')
17. player\_file.write("DROP TABLE IF EXISTS Player; \nCREATE TABLE Player (Player\_id INT, Name VARCHAR(20), "
18. "Age INT, Overall INT, NTC VARCHAR(3));\n\n")
20. **for** i **in** range(personal\_total\_rows):
21. temp = ""
22. **for** key, value **in** country\_dic.items():
23. **if** personal\_data["Nationality"][i] == key:
24. temp = value
25. player\_file.write("INSERT into Player values (" + str(personal\_data["ID"][i]) + ", " + '"''"'
26. + str(personal\_data["Name"][i]) + '"' + ", " + str(personal\_data["Age"][i]) + ", "
27. + str(personal\_data["Overall"][i]) + ", " + '"''"' + temp + '"' + ");\n")
29. player\_file.close()

(player\_preferred\_extraction.py)

1. **import** pandas as pd
2. player\_position\_data = pd.read\_csv('PlayerPlayingPositionData.csv')
4. total\_rows = player\_position\_data.shape[0]
6. f = open("Preferred.sql","w")
8. **print**("DROP TABLE IF EXISTS Preferred; \nCREATE TABLE Preferred (Player\_id INT, Preferred\_Position VARCHAR(20));\n", file=f)
10. **for** i **in** range(0,total\_rows):
11. **print**("INSERT into Preferred values (", player\_position\_data["ID"][i], ", ", '"''"', player\_position\_data["Preferred Positions"][i], '"', ");",sep="", file=f)
12. f.close()

(player\_Value\_extraction.py)

1. **import** pandas as pd
3. personal\_data = pd.read\_csv('PlayerPersonalData.csv')
5. total\_rows = personal\_data.shape[0]
7. #print(personal\_data["Wage"])
8. f = open("Value.sql","w")
9. **print**("DROP TABLE IF EXISTS Value; \nCREATE TABLE Value (Player\_id INT, Player\_Value VARCHAR(10), Player\_Wage VARCHAR(10));\n", file=f)
11. #print(total\_rows)
12. **for** i **in** range(0,total\_rows):
13. **print**("INSERT into Value values (", personal\_data["ID"][i], ", ", '"''"', personal\_data["Value"][i], '"', ", ", '"''"', str(personal\_data["Wage"][i]), '"', ");",sep="", file=f)
14. f.close()

(skills\_attack\_extraction.py)

1. **import** pandas as pd
3. #reading in raw data through pandas read\_csv command
4. playing\_pos\_data = pd.read\_csv('PlayerPlayingPositionData.csv')
5. **print**(playing\_pos\_data)
7. #extract total number of rows for iteration
8. total\_rows = playing\_pos\_data.shape[0]
10. #create new .sql file to run on sql to insert table
11. f = open("attack.sql","w")
12. **print**("DROP TABLE IF EXISTS Attack; \nCREATE TABLE Attack(Player\_id INT, CF INT, LF INT, LS INT, LW INT, RF INT, RS INT, RW INT, ST INT);\n", file=f)
14. **for** i **in** range(0,total\_rows):
15. **print**("INSERT into Attack values (", playing\_pos\_data["ID"][i], ", ", playing\_pos\_data["CF"][i], ", ", playing\_pos\_data["LF"][i], ", ", playing\_pos\_data["LS"][i], ", ", playing\_pos\_data["LW"][i], ", ", playing\_pos\_data["RF"][i], ", ", playing\_pos\_data["RS"][i], ", ", playing\_pos\_data["RW"][i], ", ", playing\_pos\_data["ST"][i], ");",sep="", file=f)
17. f.close()

(skills\_defense\_extraction.py)

1. **import** pandas as pd
3. #reading in raw data through pandas read\_csv command
4. playing\_pos\_data = pd.read\_csv('PlayerPlayingPositionData.csv')
5. #print(playing\_pos\_data)
7. #extract total number of rows for iteration
8. total\_rows = playing\_pos\_data.shape[0]
10. #create new .sql file to run on sql to insert table
11. f = open("defense.sql","w")
12. **print**("DROP TABLE IF EXISTS Defense; \nCREATE TABLE Defense(Player\_id INT, CB INT, LB INT, LCB INT, LWB INT, RB INT, RCB INT, RWB INT);\n", file=f)
14. **for** i **in** range(0,total\_rows):
15. **print**("INSERT into Defense values (", playing\_pos\_data["ID"][i], ", ", playing\_pos\_data["CB"][i], ", ", playing\_pos\_data["LB"][i], ", ", playing\_pos\_data["LCB"][i], ", ", playing\_pos\_data["LWB"][i], ", ", playing\_pos\_data["RB"][i], ", ", playing\_pos\_data["RCB"][i], ", ", playing\_pos\_data["RWB"][i], ");",sep="", file=f)
17. f.close()

(skills\_midfield\_extraction.py)

1. **import** pandas as pd
3. #reading in raw data through pandas read\_csv command
4. playing\_pos\_data = pd.read\_csv('PlayerPlayingPositionData.csv')
5. #print(playing\_pos\_data)
7. #extract total number of rows for iteration
8. total\_rows = playing\_pos\_data.shape[0]
10. #create new .sql file to run on sql to insert table
11. f = open("midfield.sql","w")
12. **print**("DROP TABLE IF EXISTS Midfield; \nCREATE TABLE Midfield(Player\_id INT, CAM INT, CDM INT, CM INT, LAM INT, LCM INT, LDM INT, LM INT, RAM INT, RCM INT, RDM INT, RM INT);\n", file=f)
14. **for** i **in** range(0,total\_rows):
15. **print**("INSERT into Midfield values (", playing\_pos\_data["ID"][i], ", ", playing\_pos\_data["CAM"][i], ", ", playing\_pos\_data["CDM"][i], ", ", playing\_pos\_data["CM"][i], ", ", playing\_pos\_data["LAM"][i], ", ", playing\_pos\_data["LCM"][i], ", ", playing\_pos\_data["LDM"][i], ", ", playing\_pos\_data["LM"][i], ", ", playing\_pos\_data["RAM"][i], ", ", playing\_pos\_data["RCM"][i], ", ", playing\_pos\_data["RDM"][i], ", ", playing\_pos\_data["RM"][i],");",sep="", file=f)
17. f.close()

**Representative sample of SQL files used to populate database (only a portion of the code is shown due to length)**

(attack.sql)

1. #Populate database with Attack(offensive) skills relation
2. DROP TABLE IF EXISTS Attack;
3. CREATE TABLE Attack(Player\_id INT, CF INT, LF INT, LS INT, LW INT, RF INT, RS INT, RW INT, ST INT);
4. INSERT into Attack values(20801, 91.0, 91.0, 92.0, 91.0, 91.0, 92.0, 91.0, 92.0);
5. INSERT into Attack values(158023, 92.0, 92.0, 88.0, 91.0, 92.0, 88.0, 91.0, 88.0);
6. INSERT into Attack values(190871, 88.0, 88.0, 84.0, 89.0, 88.0, 84.0, 89.0, 84.0);
7. INSERT into Attack values(176580, 88.0, 88.0, 88.0, 87.0, 88.0, 88.0, 87.0, 88.0);

(attribute.sql)

1. #populate database with Attributes relation
2. DROP TABLE IF EXISTS Attributes;
3. CREATE TABLE Attributes(ID INT, Acceleration INT, Aggression INT, Agility INT, Balance INT, Ball\_control INT, Composure INT, Crossing INT, Curve INT, Dribbling INT, Finishing INT, Free\_kick\_accuracy INT, GK\_diving INT, GK\_handling INT, GK\_kicking INT, GK\_positioning INT, GK\_reflexes INT, Heading\_accuracy INT, Interceptions INT, Jumping INT, Long\_passing INT, Long\_shots INT, Marking INT, Penalties INT, Positioning INT, Reactions INT, Short\_passing INT, Shot\_power INT, Sliding\_tackle INT, Sprint\_speed INT, Stamina INT, Standing\_tackle INT, Strength INT, Vision INT, Volleys INT);
4. INSERT into Attributes values(20801, 89, 63, 89, 63, 93, 95, 85, 81, 91, 94, 76, 7, 11, 15, 14, 11, 88, 29, 95, 77, 92, 22, 85, 95, 96, 83, 94, 23, 91, 92, 31, 80, 85, 88);
5. INSERT into Attributes values(158023, 92, 48, 90, 95, 95, 96, 77, 89, 97, 95, 90, 6, 11, 15, 14, 8, 71, 22, 68, 87, 88, 13, 74, 93, 95, 88, 85, 26, 87, 73, 28, 59, 90, 85);
6. INSERT into Attributes values(190871, 94, 56, 96, 82, 95, 92, 75, 81, 96, 89, 84, 9, 9, 15, 15, 11, 62, 36, 61, 75, 77, 21, 81, 90, 88, 81, 80, 33, 90, 78, 24, 53, 80, 83);
7. INSERT into Attributes values(176580, 88, 78, 86, 60, 91, 83, 77, 86, 86, 94, 84, 27, 25, 31, 33, 37, 77, 41, 69, 64, 86, 30, 85, 92, 93, 83, 87, 38, 77, 89, 45, 80, 84, 88);

(club.sql)

1. #Populate database with club relation
2. DROP TABLE IF EXISTS Club;
3. CREATE TABLE Club(Club\_id INT, Club\_name VARCHAR(40));
4. INSERT into Club values(0, "");
5. INSERT into Club values(1, "CD Leganés");
6. INSERT into Club values(2, "BSC Young Boys");
7. INSERT into Club values(3, "River Plate");
8. INSERT into Club values(4, "Chicago Fire Soccer Club");
9. INSERT into Club values(5, "Central Coast Mariners");
10. INSERT into Club values(6, "SK Sturm Graz");
11. INSERT into Club values(7, "FC Basel");
12. INSERT into Club values(8, "Vitória Setúbal");

(country.sql)

1. #sql code used to populate the Country relation
2. DROP TABLE IF EXISTS Country;
3. CREATE TABLE Country(NTC VARCHAR(3), Country\_Name VARCHAR(20), Capital VARCHAR(20), Continent VARCHAR(2), Development VARCHAR(20), Sub\_region VARCHAR(35));
4. INSERT into Country values("AFG", "Afghanistan", "Kabul", "AS", "Developing", "Southern Asia");
5. INSERT into Country values("ALB", "Albania", "Tirana", "EU", "Developed", "Southern Europe");
6. INSERT into Country values("ALG", "Algeria", "Algiers", "AF", "Developing", "Northern Africa");
7. INSERT into Country values("ANG", "Angola", "Luanda", "AF", "Developing", "Sub-Saharan Africa");
8. INSERT into Country values("ATG", "Antigua & Barbuda", "St. John's", "NA", "Developing", "Latin America and the Caribbean");
9. INSERT into Country values("ARG", "Argentina", "Buenos Aires", "SA", "Developing", "Latin America and the Caribbean");
10. INSERT into Country values("ARM", "Armenia", "Yerevan", "AS", "Developing", "Western Asia");
11. INSERT into Country values("AUS", "Australia", "Canberra", "OC", "Developed", "Australia and New Zealand");
12. INSERT into Country values("AUT", "Austria", "Vienna", "EU", "Developed", "Western Europe");

(defense.sql)

1. DROP TABLE IF EXISTS Defense;
2. CREATE TABLE Defense(Player\_id INT, CB INT, LB INT, LCB INT, LWB INT, RB INT, RCB INT, RWB INT);
3. INSERT into Defense values(20801, 53.0, 61.0, 53.0, 66.0, 61.0, 53.0, 66.0);
4. INSERT into Defense values(158023, 45.0, 57.0, 45.0, 62.0, 57.0, 45.0, 62.0);
5. INSERT into Defense values(190871, 46.0, 59.0, 46.0, 64.0, 59.0, 46.0, 64.0);
6. INSERT into Defense values(176580, 58.0, 64.0, 58.0, 68.0, 64.0, 58.0, 68.0);

(midfield.sql)

1. DROP TABLE IF EXISTS Midfield;
2. CREATE TABLE Midfield(Player\_id INT, CAM INT, CDM INT, CM INT, LAM INT, LCM INT, LDM INT, LM INT, RAM INT, RCM INT, RDM INT, RM INT);
3. INSERT into Midfield values(20801, 89.0, 62.0, 82.0, 89.0, 82.0, 62.0, 89.0, 89.0, 82.0, 62.0, 89.0);
4. INSERT into Midfield values(158023, 92.0, 59.0, 84.0, 92.0, 84.0, 59.0, 90.0, 92.0, 84.0, 59.0, 90.0);
5. INSERT into Midfield values(190871, 88.0, 59.0, 79.0, 88.0, 79.0, 59.0, 87.0, 88.0, 79.0, 59.0, 87.0);
6. INSERT into Midfield values(176580, 87.0, 65.0, 80.0, 87.0, 80.0, 65.0, 85.0, 87.0, 80.0, 65.0, 85.0);

(player.sql)

1. DROP TABLE IF EXISTS Player;
2. CREATE TABLE Player(Player\_id INT, Name VARCHAR(20), Age INT, Overall INT, NTC VARCHAR(3));
3. INSERT into Player values(20801, "Cristiano Ronaldo", 32, 94, "POR");
4. INSERT into Player values(158023, "L. Messi", 30, 93, "ARG");
5. INSERT into Player values(190871, "Neymar", 25, 92, "BRA");
6. INSERT into Player values(176580, "L. Suárez", 30, 92, "URU");
7. INSERT into Player values(167495, "M. Neuer", 31, 92, "GER");
8. INSERT into Player values(188545, "R. Lewandowski", 28, 91, "POL");

(plays\_in.sql)

1. DROP TABLE IF EXISTS Plays\_in;
2. CREATE TABLE Plays\_in(Player\_id INT, Club\_id INT);
3. INSERT into Plays\_in values(20801, 349);
4. INSERT into Plays\_in values(158023, 332);
5. INSERT into Plays\_in values(190871, 326);
6. INSERT into Plays\_in values(176580, 332);
7. INSERT into Plays\_in values(167495, 501);

(Position.sql)

1. DROP TABLE IF EXISTS Positions;
2. CREATE TABLE Positions(Position\_Abv VARCHAR(4), Position\_Name VARCHAR(25));
3. INSERT INTO Positions values("CAM", "Center\_Attacking\_Midfield");
4. INSERT INTO Positions values("CB", "Center\_Back");
5. INSERT INTO Positions values("CDM", "Center\_Defensive\_Midfield");
6. INSERT INTO Positions values("CF", "Center\_Forward");
7. INSERT INTO Positions values("CM", "Center\_Midfield");
8. INSERT INTO Positions values("LAM", "Left\_Attacking\_Midfield");
9. INSERT INTO Positions values("LB", "Left\_Back");
10. INSERT INTO Positions values("LCB", "Left\_Center\_Back");
11. INSERT INTO Positions values("LCM", "Left\_Center\_Midfield");
12. INSERT INTO Positions values("LDM", "Left\_Defensive\_Midfield");
13. INSERT INTO Positions values("LF", "Left\_Forward");
14. INSERT INTO Positions values("LM", "Left\_Midfielder");

(Preferred.sql)

1. DROP TABLE IF EXISTS Preferred;
2. CREATE TABLE Preferred(Player\_id INT, Preferred\_Position VARCHAR(20));
3. INSERT into Preferred values(20801, "ST LW ");
4. INSERT into Preferred values(158023, "RW ");
5. INSERT into Preferred values(190871, "LW ");
6. INSERT into Preferred values(176580, "ST ");
7. INSERT into Preferred values(167495, "GK ");
8. INSERT into Preferred values(188545, "ST ");
9. INSERT into Preferred values(193080, "GK ");
10. INSERT into Preferred values(183277, "LW ");
11. INSERT into Preferred values(182521, "CDM CM ");
12. INSERT into Preferred values(167664, "ST ");
13. INSERT into Preferred values(155862, "CB ");

(Value.sql)

1. DROP TABLE IF EXISTS Value;
2. CREATE TABLE Value(Player\_id INT, Player\_Value VARCHAR(10), Player\_Wage VARCHAR(10));
3. INSERT into Value values(20801, "€95.5M", "€565K");
4. INSERT into Value values(158023, "€105M", "€565K");
5. INSERT into Value values(190871, "€123M", "€280K");
6. INSERT into Value values(176580, "€97M", "€510K");
7. INSERT into Value values(167495, "€61M", "€230K");
8. INSERT into Value values(188545, "€92M", "€355K");
9. INSERT into Value values(193080, "€64.5M", "€215K");
10. INSERT into Value values(183277, "€90.5M", "€295K");

Phase I

Ye Chan Kim

Jae Goan Park

1. **List of Potential English Questions**
   1. Which player has the highest “Overall” Stat? (List all ties)
   2. List the nationality and ID of all players with the highest wages.
   3. List the club name and average wage of players by soccer club name.
   4. List all the teams that are located in Spain.
   5. List all players with “LB (Left Back)” stat higher than total LB stat average of all players.
   6. List the player who is a striker and has the highest potential to be a goal keeper.
   7. List players with more than one preferred position.
   8. List the top ten strongest players in the FIFA 18 database.
   9. List the 5 countries with the respective players that have the highest chance of scoring in a penalty kick
   10. List the most valuable club in the database.
   11. At what age do players have the highest overall average?
   12. List the fastest and the slowest player on the database.
   13. List the player name, age, and offensive stats for “Real Madrid” players
   14. Make the best team possible for a 4-3-3 formation where the players are from Spain.
   15. Who would most likely win between Real Madrid and Barcelona based on player stats.
2. Relational Model

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Player** | Player ID | Name | Nationality | Age | Team | Overall | |  | 1179 | G. Buffon | Italy | 39 | Juventus | 89 | |  |  |  |  |  |  |  | | **Offensive Stat** | Player ID | Finishing | Dribbling | Agility | Acceleration |  | |  | 155862 | 80 | 90 | 76 | 89 |  | |  |  |  |  |  |  |  | | **Defensive Stat** | Player ID | Aggression | Interception | Strength |  |  | |  | 155462 | 40 | 78 | 95 |  |  | |  |  |  |  |  |  |  | | **Team** | Team Name | Country | Continent |  |  |  | |  | Real Madrid | Spain | Europe |  |  |  | |  |  |  |  |  |  |  | | **Continent** | Continent Name | Country Name | GDP (USD Tn) | Life Expectancy | Crime Rate Ranking |  | |  | Asia | Japan | 4.9 | 83 | 12 |  | |  |  |  |  |  |  |  | | **Finance** | Player ID | Wage (Mn) | Net Worth (mn) |  |  |  | |  | 1179 | 4 | 50 |  |  |  | |  |  |  |  |  |  |  | | **Position** | Player ID | Preferred Position |  |  |  |  | |  | 1179 | ST |  |  |  |  | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

v) SQL Statements

Make the best team possible for a 4-3-3 (plus goal keeper) formation where the players are from Spain.

SELECT P.playerid

FROM Position as O,

(SELECT P.playerid

FROM Player as P

WHERE P.nationality = “Spain”) as S1

WHERE P.playerid = O.playerId AND O.preferredposition = “ST” AND

P.overall = (SELECT MAX(Overall)

FROM Player as P, Position as O

WHERE P.playerid = O.playerId AND O.preferredposition = “ST”);

\*NOTE: Repeat this procedure 11 times for all the other positions involved in the 4-3-3 formation (e.g. LW, ST, RW, LM, CM, RM, LB, CB, CB, RB, GK) and put together which a stored SQL procedure so output is integrated as one answer.

vi) Plan for loading database

We will download a comprehensive FIFA 18 player stat data from Kaggle.com from <https://www.kaggle.com/thec03u5/fifa-18-demo-player-dataset>. We have not found appropriate database for country/continent stats, but are likely to collect from UN or CIA World Factbook website. The data collected will be in form of csv.

Some issues we have noticed are that some team names and player names display invalid broken text. This is probably because the original name or team name contains special characters that cannot be recognized in csv format. We will attempt to find a way to convert all of them to standard English so they are readable. Hopefully we do not have to convert them by hand one by one.

vii) Type of result we wish to generate

We wish to provide hardcore FIFA 18 gamers with the most optimal set of players to use for creating their team. Users of this database, will be able to input their general preferences about their dream team and be able to obtain several viable dream team options.

viii) Specialized/advanced topics

We plan on including advanced SQL topics during project implementation. To create the best team formation different combinations of player formations need to be formatted in a comprehensive way.