Alexandra Flores

Gage Anderson

Omar Zaher

Sajid Anjum

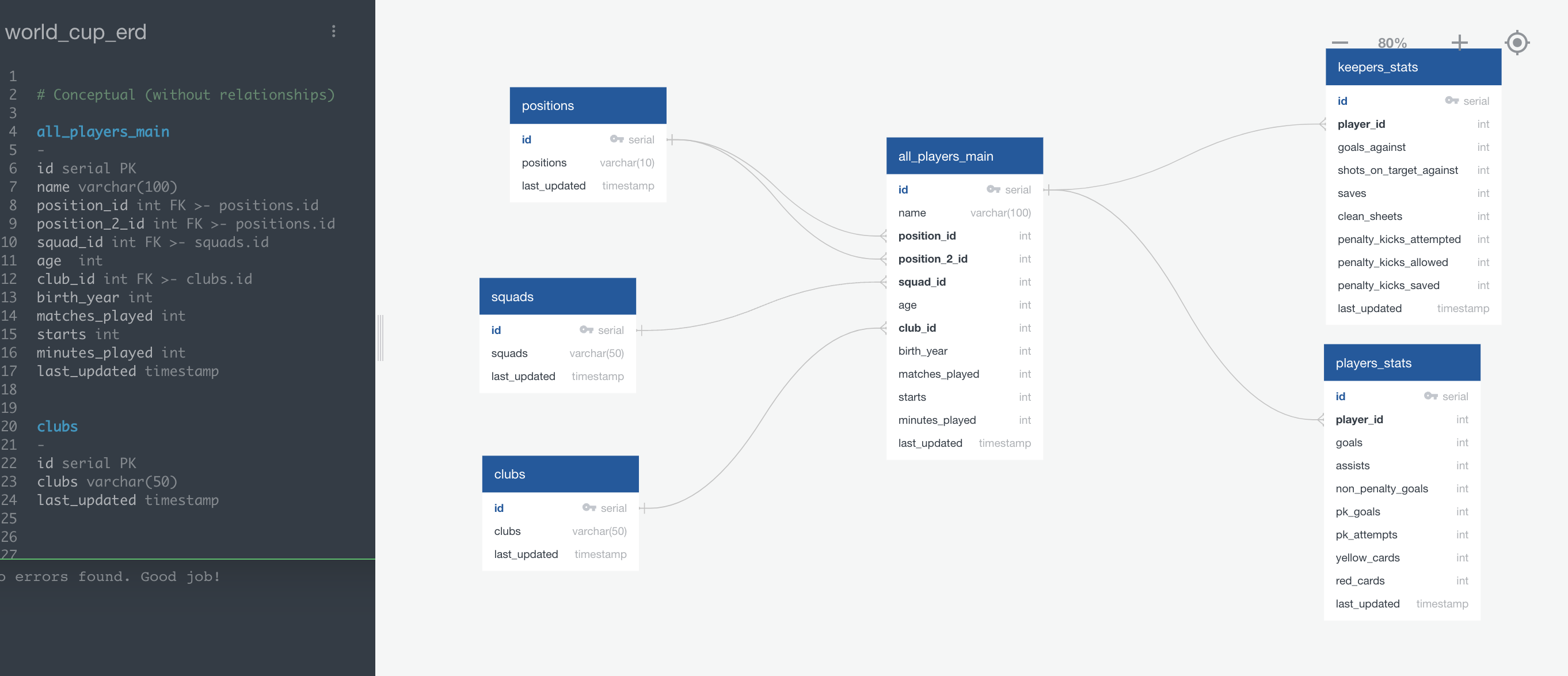
**Project 2- ETL-World Cup Write up**

Since the world cup is currently in the news, we would like to take a deeper dive into the different players, teams, and stats of the FIFA World Cup 2022. This tournament has happened once every 4 years for the past 92 years. There is plenty of data for us to use and analyze; at the same time we hope this analysis will give us better knowledge about soccer or “football” and whether the number of goals scored correlates with making it further in the cup. We are all very interested in this topic since it rarely takes center stage, and who doesn’t love the World Cup?!

We managed to find two URL’s that we web scraped, and we specified “players\_stats”, “keepers\_stats”, “positions”, “clubs”, “squads” (National teams), and “all\_players\_main” as our 6 categorical tables. Since we had 2 URL’s, one for players and one for goal keepers, we had to create a junction table that combined the data from the players URL with the data from the keepers URL. We labeled this table as “all\_players\_main”, and this table contains basic information about each player such as name, age, nation, club etc. The “players\_stats” and “keepers\_stats” tables link the players statistics to the player information in the main table. Lastly, the “clubs”, “squads”, and “positions” table store categorial data about the clubs, squads, and positions associated with each player and also connect to the main table. We will b be using the data in these tables to sort and group by different stats like goals, assists, and playing time. Doing this would allow us to analyze whether goals scored helps improves longevity in the cup.

**Constructing the ERD**:

Our next step was to create the ERD using quickdatabasediagrams.com using the columns from the parent tables in the URLs. One of the first things we realized was that some players were assigned more than one position in one single column named “position\_id”, so we decided to make a new column named “position\_2\_id” in our main table to separate the second position. In our main table we decided to name our columns “ id”, “name”, “position\_id”, “position\_2\_id”, “squad\_id”, “age”, “club\_id”, “birth\_year”, “matches\_played”, “starts”, “minutes\_played”, and “last updated (as a time stamp)””. We then created the 5 other tables with id being a serial primary key for each table. The foreign keys on our main table were, “position\_id”, “position\_2\_id”, “squad\_id”, and “club\_id”. We linked position\_id and “position\_2\_id” as foreign keys linked to id in the positions table. Similarly, we did the same with “squad\_id” and “club\_id” with foreign keys linked to the serial id in each respective table. Lastly, we linked “player\_id” in the “keepers\_stats” table and “player\_id” in the “players\_stats” table to link as foreign keys to the “all\_players\_main” table. The result is depicted below.



**Extract:**

**Transform:**

**Load:**

To load the data onto Postgres we used Jupyter Notebook to upload the schema and the different CSV dataframes. Once we did this we were able to view all the different dataframes and create queries that we found to be useful to create an analysis on the World Cup. We were able to pull information based on

For the web scraping, we chose to scrape the table “**Player Standard Stats 2022 World Cup**” from the URL <https://fbref.com/en/comps/1/stats/World-Cup-Stats> and the table “**Player Goalkeeping 2022 World Cup**” <https://fbref.com/en/comps/1/keepers/World-Cup-Stats> from the URL on the [fbref.com](https://d.docs.live.net/38686bf526439c97/Documents/fbref.com) website. Because the data loaded in chunks and not all at once, we had to use Selenium, Splinter, and the Chrome Driver to use the dummy Chrome browser to scrape after the data had been rendered on the webpage. Since the data is already in a table format, we were able to use BeautifulSoup to extract the table html and use the pd.read\_html() method to convert that html to a pandas dataframe. This caused an extra row of grouped headers to be included (ex: ( 'Unnamed: 0\_level\_0', 'Rk')), so we renamed the columns. Additionally, every 25th value was another header column, so we used a for loop to drop every 25th row in each of the players and keepers tables. Finally, the data frames were saved off as the csv files “players.csv” and “keepers.csv”. This process can be found in the Jupyter notebook “WorldCupScraping”.

**Transforming the Data:**

**Step 1: Creating the “all\_players\_main” table.**

As mentioned in the **ERD** section, our first goal was to change the original two tables into a player information table. To do this, we extracted the columns:

“name”, “position\_id”, “position\_2\_id”, “squad\_id”, “age”, “club\_id”, “birth\_year”, “matches\_played”, “starts”, “minutes\_played”

The “id” column was automatically generated by PostGres. The “name” column was categorical player information, but since very player has a unique name, we did not see a use to make a “name” table. “position”, “squad”, and “club” are categorical variables, so we made unique tables for those variables and replaced them with the primary keys of those respective tables. All the other columns consist of quantitative data, so we did not modify that.

In order to replace “squad” with “squad\_id”, we made a dictionary with the unique squads as keys and a serial list of numbers starting from 1 as the values. The .loc function over a for loop was used to do the replacements:

squads\_dict **=** {}

**for** x **in** range(len(df\_squads)):

squads\_dict[df\_squads**.**squads**.**tolist()[x]] **=** x**+**1

**for** i **in** range(len(dftp)):

dftp**.**iloc[i, 3] **=** squads\_dict[dftp**.**iloc[i, 3]]

We repeated this same process for clubs and positions.

We also noticed that players and keepers have different stats associated with them, so we made separate “keepers\_stats” and “players\_stats” tables and connected the “name” column in those tables with the primary key in the “all\_players\_main” table. We used the same dictionary process to replace the “name” column with the “player\_id” column

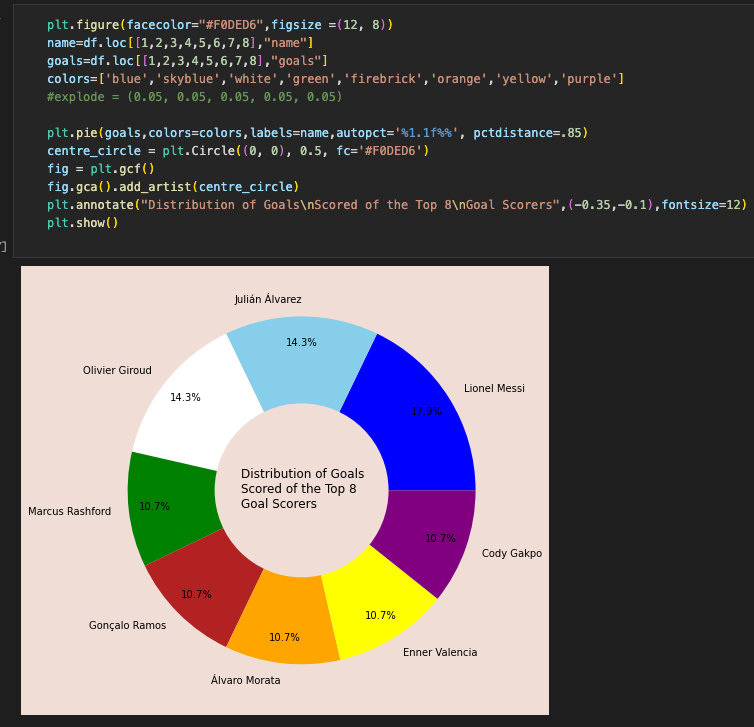
**Analysis**:

For our analysis we wanted to see the total goals by squad (national team) to help determine which squads scored the most. As it turns out France and Argentina were top contenders for scoring which makes sense since they made it to the final match. An outlier would be Brazil, they were one of the top scorers but only made it to the quarter finals.

Background pattern

Description automatically generated

We then looked at Goals by top 8 players to see who the top scorers were and if they lined up with the squads. The data showed that Messi and Mbappe were the top scorers which lines up with both of their squads making it to the finals. In conclusion, it seems if you made over 4 goals more than likely your squad would have made it to the quarter finals at the least and then anything over 5 would lead to a higher probability of making it to the finals.



**Future work and limitations:** In the future we would like to find a salaries table through a URL somewhere online for all the soccer players and scrape that data so that we could see if there are any players who are over/under valued. We could also have improved our data base and linked the data for clubs with squads (national team). We also could have pulled all the World Cup information all the way up to the final since we had the time.

**Automation:** We searched a little bit online to automate this process, and the easiest way seems to combine all our code into a single Jupyter Notebook cell and export the code to a python editor. The three notebooks could be defined as three different python methods, one to scrape the webpages, one to transform the data, and the third to export the database. All of the three methods would be called from a main .py file within a for loop. The length of the for loop would be the number of days you want to run the code, and at the end of very for loop, we could call the time.sleep() method for 86400 seconds which is the length of one day. Thus, if we wanted the file to run at 2:00 am every night, and we pressed the run key at 5:00 PM before leaving the office, we would do an initial sleep for 32400 seconds (9 hours), and then the for loop would upload all the data to a PostGres database every night at 2:00 am automatically for a pre-specified number of days. This is very inefficient as it recreates the entire database every night, but for our purposes, I don’t see why it should be a problem since our dataset is pretty small. For larger datasets, we could use upsert instead of insert for a combined update and insert in order to save memory.

Works cited

1. Field Player Stats:

https://fbref.com/en/comps/1/stats/World-Cup-Stats

1. Goal Keeping:

<https://fbref.com/en/comps/1/keepers/World-Cup-Stats>

1. ERD:

https://app.quickdatabasediagrams.com