Homework 7: Databases

In this homework, you will be populating a database table with information about footballers and then writing code to fetch data from that table. You will need the starter code, called HW7.py, and the players file, called football.json.

In case you aren't familiar: football, or soccer, is the most popular sport in the world. The English Premier League is the most-watched competition and features some of the best players in the world at any given time. It doesn't always feature the same teams, as teams that the lowest-performing teams are relegated at the end of each season to a lower league; they are replaced by the best teams from below. The data you'll use in this assignment is from **football-data.org**; this is an example of the type of APIs you'll access for your final project.

We have provided the code for the following:

1. To read the cache data (***read_data()*** function)
2. To create the database and set up the connection and cursor (***open_database()*** function)
3. To set up one of the tables, called Positions, in the database (***make_positions_table()*** function)

We have also provided test cases that will pass if the functions are written correctly. **You should not edit these test cases.** NOTE: It is okay for the extra credit test case to fail if you do not attempt the extra credit; you can also comment out those specific test cases.

When done with the assignment, your database will have two tables. You should start by running the starter code and looking at the structure of the Positions table in DB Browser.

Tasks

1. ***make_players_table()***

This function takes 3 arguments: JSON data, the database cursor, and the database connection object. It iterates through the JSON data to get a list of players in the squad and loads them into a database table called 'Players' with the following columns:

- id ((datatype: int; Primary key) - note this comes from the JSON
- name (datatype: text)
- position_id (datatype: integer)
- birthyear (datatype: int)
- nationality (datatype: text)

To find the position_id for each player, you will have to look up their position in the Positions table we created for you -- see make_positions_table above for details.

This is what your Players table should look like when viewed in DB Browser:

	id	name	position_id	birthyear	nationality
	Filter	Filter	Filter	Filter	Filter
1	146	Jadon Sancho	2	2000	England
2	3188	David De Gea	0	1990	Spain
3	3231	Casemiro	2	1992	Brazil
4	3232	Fred	2	1993	Brazil
5	3257	Bruno Fernandes	2	1994	Portugal
6	3308	Jack Butland	0	1993	England
7	3326	Harry Maguire	1	1993	England
8	3331	Marcus Rashford	3	1997	England
9	3360	Raphaël Varane	1	1993	France
10	3372	Anthony Martial	3	1995	France
11	3459	Christian Eriksen	2	1992	Denmark

2. ***nationality_search(countries, cur, conn)***

This function takes 3 arguments as input: a list of countries, the database cursor, and database connection object. It selects all the players from any of the countries in the list and returns a list of tuples. Each tuple contains the player's name, their position_id, and their nationality.

3. ***birthyear_nationality_search(age, country, cur, conn)***

This function takes 4 arguments as input: an age in years (int), a country (string), the database cursor, and the database connection object. It selects all the players from the country passed to the function that were born ***BEFORE*** (2023 minus the year passed). For example: if we pass 19 for the year, it should return players with birthdates ***BEFORE*** 2004. This function returns a list of tuples each containing the player's name, nationality, and birth year.

4. ***position_birth_search(position, age, cur, conn)***

This function takes 4 arguments as input: a position (string), age (int), the database cursor, and the database connection object. It selects all the players who play the position passed to the function and that were born ***AFTER*** (2023 minus the year passed). for example: if we pass 19 for the year, it should return players with birth years ***AFTER*** 2004. This function returns a list of tuples each containing the player's name, position, and birth year.

HINT: You'll have to use JOIN for this task.

5. **Extra Credit**
   Add args to parens

   You'll make 3 new functions, ***make_winners_table()***, ***make_seasons_table()***, and ***winners_since_search()***, and then write at least 2 meaningful test cases for each of them.

   The first function takes 3 arguments: JSON data, the database cursor, and the database connection object. It makes a table with 2 columns:
   - id (datatype: int; Primary key) -- note this comes from the JSON
   - name (datatype: text) -- note: use the full, not short, name

   HINT: look at how we made the Positions table above for an example

   The second function takes the same 3 arguments: JSON data, the database cursor, and the database connection object. It iterates through the JSON data to get info about previous Premier League seasons (don't include the current one) and loads all of the seasons into a database table called 'Seasons' with the following columns:
   - id (datatype: int; Primary key) - note this comes from the JSON
   - winner_id (datatype: text)
   - end_year (datatype: int)

   To find the winner_id for each season, you will have to look up the winner's name in the Winners table. NOTE: Skip seasons with no listed winner!

   The third function takes in a year (string), the database cursor, and the database connection object. It returns a dictionary of how many times each team has won the Premier League since the passed year. In the dict, each winning team's (full) name is a key, and the value associated with each team is the number of times they have won since the year passed, including the season that ended the passed year.

Grading Rubric

1. ***make_players_table()*** - 25 points
   a. 5 points for making a table
   b. 5 points for adding all values correctly
   c. 15 points for filling table with players

2. ***nationality_search()*** - 10 points

3. ***birthyear_nationality_search()*** - 10 points

4. ***position_birth_search()*** - 15 points

5. **Extra credit** - 6 bonus points

Submission

Make at least 4 git commits and turn in your GitHub repo URL on Canvas by the due date to receive credit.