Siyabonga Mabuza
Fares Yahmadi
12/05/2024

Documentation for how to insert a large amount of data into your database and how to run your API.

In order to insert a large amount of data into our soccer database, we opted to write a python script using libraries such as Faker, MariaDb and Random. This allowed us to control the different values of our attributes, making the data more realistic and also making data insertion much more efficient.
If you haven't installed the faker, mariadb and random library, run this command.

Pip install faker mariadb random

For every table in our Soccer Database which are: Player, Team, League, Transfers we built a generate function that creates data for our attributes and later Inserts those values into their respective tables. Before we discuss the functions that help generate the data for the Soccer database, it is important to note:

There are 5 leagues.
Every league has between 18 to 20 teams.
In total we have 96 teams.
Every team contains 30 players: 27 players and 3 goalkeepers.
Therefore we need to generate 96*30 = 2,880 players for our database.
600 of those players have a transfer history.

In the generator file we have 5 different functions that populate the database in order to be able to test the queries. These functions are the following:

1- `def generate_players(x,db_cur,team_id):`

```
—-> Generates 27 players with infield positions' (Attacker, Midfielder,
Defender) attributes using the faker and random library and inserts
them into the player table.
      x : is a loop counter that increases by 30 to insert 30 players
      per team
      Db_cur : cursor object
      Team_id: corresponds to the id of the team we are inserting
      players into
```

2 - `def generate_goalkeepers(x,db_cur, team_id):`

```
—-> Generates 3 goalkeeper attributes and inserts them into the player
table.
      x : is a loop counter that increases by 30 to insert 30 players
      per team
      Db_cur : cursor object
      Team_id: corresponds to the id of the team we are inserting the
      goalkeepers into
```

Siyabonga Mabuza
Fares Yahmadi
12/05/2024

3- `def generate_leagues(leagues_array, db_cur):`
```
—-> Generates leagues attributes and inserts the data into the league
table .
      League_array : list of tuples containing the league ID, league
      name and league country.
      db_cur  : cursor object
```

4- `def generate_team(teams, team_id, league_id,db_cur):`
```
—-> Generates team attributes and inserts them into the team table.
      Teams: list of teams from a specific league
      Team_id: integer that increments by 1 to assign each team a
      unique ID. 100 for German teams, 200 for French teams etc ..

      League_id : every team is assigned a league_id which indicates
      which league it belongs too.
```

5- `def generate_transfer(x,db_cur):`
```
—-> Generates transfers history attributes and inserts them into the
transfers table.
      x : number of transfers being generated. In our case x = 600.
      Db_cur: cursor object.
```

Main function:
In the main, these 5 different functions will be called to generate the data in question.
1 - We call the generate_league function by passing leagues array as a parameter. This generates 5 leagues with unique IDs and inserts them in the leagues table.
2- Using the get_ids_from_db function we are able to access all the IDs of the different leagues we just inserted. We iterate through those IDs to generate their respective teams and assign them unique IDs that correspond to the league. The generate_team function also inserts this data into the team table.
3- Once we have created the league and team table we can begin to add 30 players: 27 infield players and 3 goalkeepers to each team. For that we iterate through all the team ids we have just inserted, and call the generate_goalkeeper and generate_players functions. To ensure every player has a unique ID we create a count1 that increments by 30.

Siyabonga Mabuza
Fares Yahmadi
12/05/2024

API:
For our API, we ran the following test cases on our 6 queries:

Query 1 : Gets user to input a position. Returns all players who play in that given position.

Query 2 : Gets user to input a min transfer value. Returns all players who cost less than given value.

Query 3 : Gets user to input a min age and a country. Returns the players younger than the given age, who play in the given country.

Query 4 : Gets user to input a position. Returns average mins played by players in that position.

Query 5 : Gets user to input max transfer fee and min sum of goals and assists (G/A). Returns players who cost less than given transfer fee, and have more than given G/A.

Query 6 : Gets user to input min games played and position to exclude. Return players who have played more games than given input, excluding player in given position.