Phase 2
Uzair Chudhary and Elhan Syed

Design Decisions

We received full marks and positive feedback on our phase 1 from our TA. There were no suggested changes or improvements. However, upon further review of the data, we decided to make a few changes anyway to improve our database and make it usable for future reference and updates.

Firstly, we removed the *leagueID* column from the **Team** table as we realized that it was not possible to join any tables and match the teams with the league they played in based on the available data. Arguably, we could have added it manually by searching up where each team played every season but it would have been laborious and caused issues because of the promotion/relegation system in most European leagues where teams change their league based on their performances.

Next, we decided to remove the date constraints in the **Match** and **PlayerAttribute** tables because, if we wanted to update the database with contemporary statistics, it would not be possible. To future-proof our schema, we decided it was best to remove the date constraints. Lastly, to improve readability, we renamed columns according to snake-case where all words are lowercase and underscores are used to separate words.

We also reorganized the order of the tables as the schema would cause missing reference errors if **Match** (which references **Country, Team,** and, **League**) were to be created before the referred tables.

We decided to rename the *cross* column in **Match** (which refers to the number of crosses in a match) to *crosses* because ‘cross’ is a keyword in flavours of SQL and was bound to cause errors.

We added all the foreign key constraints from Phase 1 and all the check values into the schema in the relevant tables (except the aforementioned date constraints) along with adding not null checks where they were appropriate (certain columns were clearly less useful and were excluded from the not null checks).

Cleaning Process

Our cleaning process was lengthy because of the size of the datafile (over 300 MB) and the format it was in (SQLite). Firstly, we downloaded the file from https://www.kaggle.com/hugomathien/soccer and stored it in our phase2 folder on our computers. We opened an Anaconda distribution of Python with the Spyder development environment as they are both specialized for scientific computing and work with databases nicely. Then, we imported the SQLAlchemy library module for Python to deal with the input data being in SQLite format and wanting our results to be in PostgreSQL.

We created a SQLAlchemy engine with a connection pointing to the stored file. We imported the pandas library module for reading, writing, and manipulating data. We extracted each table with a "Select *" statement and our engine connection in the pandas read_sql method. The data was now in pandas dataframes, one for each table. We spliced the dataframes with square brackets that listed each of the columns we wanted from the dataframe which automatically dropped the remaining columns. After that, we used the "rename()" function to match our readability style with snake-case and to match the names we had listed in our Schema and our phase1 document. We closed the connection and disposed of it in accordance with SQLAlchemy documentation.

Then, on the side, we created a PostgreSQL database instance using the administration tool pgAdmin. We created a new SQLAlchemy connection in Python that connected to the PSQL database we had started. Using the to_sql method from pandas, we converted the dataframes one by one to tables in our PSQL database. We imported the data without applying the schema as there were issues with foreign key constraints and other race conditions. The schema was the regular 'public' schema at that point. Once the data was fully imported, we imported our own schema called 'projectschema' (which is the name of our schema in the DDL file) and did an Insert Select * From each table in public to projectschema; we removed all entries that our schema flagged as violations. Next, we set the search path to the new schema so that any new entries will be run against our created schema and deleted the public schema. Lastly, we tested the schema with numerous entries that violated specific keys or constraints or checks and the schema rejected them all correctly.

Note: The original file is a 300 MB SQLite file; we converted it to pandas dataframes and converted that to a csv to give an idea of the sample data. There was no way to copy a bit of the SQLite to another file and upload that nor could we upload the original file because of MarkUs file size limitations. We uploaded our Python code in db.py to show how we converted and transferred the data.