Assignment 3 - SQL

- Henry Woodyard

In [ ]:
```
import os
import sqlite3
soccerConnection = sqlite3.connect('database.sqlite')
cursorObj = soccerConnection.cursor()
```

Question 1

Write a SQL query that lists all the players born between 1987 and 1990 inclusive, sort them from the oldest to the youngest

In [ ]:
```
cursorObj.execute("SELECT player_name AS 'Player Name', birthday AS 'Birthday'
\
                   FROM Player \
                   WHERE birthday BETWEEN '1987-01-01' AND '1990-12-31' \
                   ORDER BY birthday")
print(list(map(lambda entry: entry[0], cursorObj.description)))
for row in cursorObj.fetchall():
    print(row)
```

Question 2

Write a SQL query that ranks all countries and leagues based on the total amount of total goals scored per game in the whole dataset. Sort them by the largest to the smallest amount of goals. Note: Read this carefully.

In [ ]:
```
cursorObj.execute("SELECT C.name AS 'Country Name', L.name AS 'League Name', S
UM(M.home_team_goal + M.away_team_goal) AS 'Total Goals Scored' \
                   FROM Match AS M, Country AS C, League AS L \
                   WHERE M.country_id = C.id AND M.league_id = L.id \
                   GROUP BY M.country_id \
                   ORDER BY `Total Goals Scored` DESC \
                   LIMIT 10;")
print(list(map(lambda entry: entry[0], cursorObj.description)))
for row in cursorObj.fetchall():
    print(row)
```

Question 3

Write a SQL query that ranks all teams by the average of all their attributes (not the players' attributes), sort them from best to worst.

In [ ]:
```
cursorObj.execute("SELECT T.team_long_name AS 'Team Long Name', AVG((TA.buildU
pPlaySpeed + TA.buildUpPlayPassing + TA.chanceCreationPassing + TA.chanceCreat
ionCrossing + TA.chanceCreationShooting + TA.defencePressure + TA.defenceAggre
ssion + TA.defenceTeamWidth)/8) AS 'Average of Attributes' \
                  FROM Team_Attributes AS TA, Team AS T \
                  WHERE T.team_api_id = TA.team_api_id \
                  GROUP BY TA.team_api_id \
                  ORDER BY `Average of Attributes` DESC;")
print(list(map(lambda entry: entry[0], cursorObj.description)))
for row in cursorObj.fetchall():
    print(row)
```

Question 4

Write a SQL query that ranks all teams by the average of their players' attributes, sort them by descending order displaying only the top 5. The output of this query should be of the form:

In [ ]:
```
cursorObj.execute("SELECT T.team_long_name AS 'Team Name', NULL AS 'Number of
 Players', AVG((TA.buildUpPlaySpeed + TA.buildUpPlayPassing + TA.chanceCreatio
nPassing + TA.chanceCreationCrossing + TA.chanceCreationShooting + TA.defenceP
ressure + TA.defenceAggression + TA.defenceTeamWidth)/8) AS 'Player Attribute
 Average' \
                  FROM Team_Attributes AS TA, Team AS T \
                  WHERE T.team_api_id = TA.team_api_id \
                  GROUP BY TA.team_api_id \
                  ORDER BY `Player Attribute Average` DESC \
                  LIMIT 5;")
print(list(map(lambda entry: entry[0], cursorObj.description)))
for row in cursorObj.fetchall():
    print(row)
```

Question 5

Write a SINGLE SQL query that finds the date that had the most goals scored on, per each different season and league.

In [ ]:
```
cursorObj.execute("SELECT STRFTIME('%d-%m-%Y', date) AS 'Date', season AS 'Sea
son', name AS 'League Name', MAX(home_team_goal + away_team_goal) AS 'Goals sc
ored' \
                  FROM Match, League AS L \
                  WHERE league_id == L.id \
                  GROUP BY Season, `League Name`")
print(list(map(lambda entry: entry[0], cursorObj.description)))
for row in cursorObj.fetchall():
    print(row)
```

Question 6

Write a SINGLE SQL query that finds the top 5 teams in terms of goals scored PER league for the 2008/2009 season.

In [ ]:
```
# The innermost nested subquery gets the goals scored per match by each team u
sing a boolean for whether it was the home team or away team.
# The next subquery ranks the teams within the leagues.
# The outside query renames variables and subsets to take only the top 5 team
s.

# The pdf version ruins the formatting. Sorry for the confusion.

cursorObj.execute(" SELECT '2008/2009' AS 'Season', L.name AS 'League', 'Ran
k', h.team_long_name AS 'Team Name',  h.goals AS 'Goals Scored' \
                   FROM \
                   ( \
                       SELECT g.team_long_name, g.league_id, g.goals, RANK()
 OVER (PARTITION BY g.league_id ORDER BY g.goals DESC) as 'Rank' \
                       FROM \
                           ( \
                           SELECT team_long_name, team_api_id, league_id, SUM
((team_api_id == home_team_api_id) * home_team_goal + (1 - (team_api_id == hom
e_team_api_id)) * away_team_goal) AS goals \
                           FROM Team, Match AS M \
                           WHERE team_api_id == home_team_api_id OR team_api_
id == away_team_api_id AND season == '2008/2009' \
                           GROUP BY team_long_name \
                           ORDER BY league_id, goals DESC \
                           ) AS g \
                    ) AS h, League AS L \
                   WHERE h.'Rank' <= 5 AND L.id == h.league_id") \

print(list(map(lambda entry: entry[0], cursorObj.description)))
for row in cursorObj.fetchall():
    print(row)
```