Relational Schema:

Person(pid, fname, lname, DOB)

Audience(pid, email) pid foreign key referencing Person(pid)

Referee(pid, country, YOE) pid foreign key referencing Person(pid)

Player(pid, shirtNum, genPos) pid foreign key referencing Person(pid)

Coach(pid, role) pid foreign key referencing Person(pid)

PlayerGameInfo(mid,pid,y1,y2,r,pos) pid foreign key referencing Player(pid), mid foreign key referencing Match(mid)

Substitution(pid1, pid2, time, mid) pid1 foreign key referencing Player(pid), pid2 foreign key referencing Player(pid), mid foreign key referencing Match(mid)

RefereeGameInfo(pid, role, mid) pid foreign key referencing Referee(pid), mid foreign key referencing Match(mid)

Team(country, officialName, URL, group, groupPoints)

GoalInfo(mid, time, forTeam, player, penalty) mid foreign key referencing Match(mid), forTeam foreign key referencing Team(country), player foreign key referencing Player(pid)

Stadium(name, location, maxCapacity)

Match(mid, LOM, startTime, date, stadium, team1, team2) stadium foreign key referencing Stadium(name), team1 foreign key referencing Team(country), team2 foreign key referencing Team(country)

Ticket(tid, price, section, range, seat, mid, stadium) mid foreign key referencing Match(mid), stadium foreign key referencing Stadium(name)

Sales(pid, tid) pid foreign key referencing Person(pid), tid foreign key referencing Ticket(tid)

PlayerTeam(pid, team) pid foreign key referencing Player(pid) team foreign key referencing Team(country)

CoachTeam(pid, team) pid foreign key referencing Coach(pid) team foreign key referencing Team(country)

Pending constraints:

From ER diagram to the ER schema:

1. A person can be both a player, a coach and a referee, which shouldn’t happen according to our ER Diagram.

Things that can be added as check constraints:

1. Yellow 1 should be true before yellow 2 card.
2. The group can only be of one of the valid options
3. The general positions of the managers, players, and coaches should be from a specific list of options
4. The match times in the same place on the same day need to have a bit of difference in time between them (couple of hours) for the match and for people to go in and out of the stadium and for cleanup
5. The times for the goals should be during the match
6. The time for the match should be between some specific time range example no more matches start after 12 am.
7. The number of tickets sold should be less or equal to the maximum capacity of the stadium.
8. The date of birth accepted should be have certain limit example not children born on the same day of the match and no person above for example 100 years old.
9. No person with more years of experience than their age or close to it.
10. Team 1 and team 2 in a match should be different.
11. The location and the stadium should exist.
12. Player substitution can only happen once (once a player is substituted out then can’t come back in).

SQL Queries:

1. with A as ( select MID

from GOALINFO G

where G.PLAYER = (

select p.PID

from PLAYER p

intersect

select p.pid

from PERSON p

where p.FNAME = 'Christine' and p.LNAME = 'Sinclair'

)

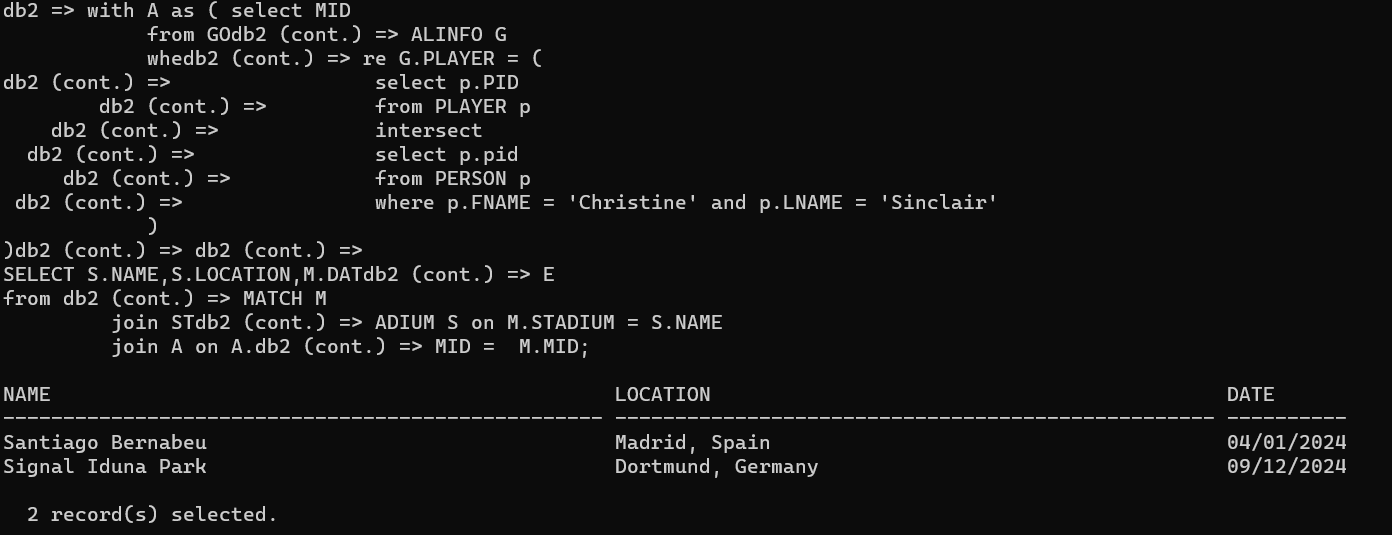
)

SELECT S.NAME,S.LOCATION,M.DATE

from MATCH M

join STADIUM S on M.STADIUM = S.NAME

join A on A.MID = M.MID;

Table A returns the match id of the goals that are scored by the players called “Christine Sinclair” this is done by getting all the people called “Christine Sinclair” then checking if their id is in the Players table. For the full query, we select the stadium name from the stadium table, stadium location from the stadium table, and match date from match date from the match table. This information is obtained by joining the match table with the stadium table on the same stadium name and then further joining the resulting table with table A on the match id to finally get all the games that the player “Christine Sinclair” has played in and scored at least one goal. 

1. with playerMatchCount as (select pid, count(\*) as mc

from PLAYERGAMEINFO

group by pid),

team1Count as (select TEAM1, count(\*) t1c

from MATCH

group by TEAM1),

team2Count as (select TEAM2, count(\*) t2c

from MATCH

group by TEAM2),

teamMatchCount as (select TEAM1, t2c + t1c as mc

from team1Count

inner join team2Count on TEAM2 = TEAM1

union

select TEAM1, t1c

from team1Count

left join team2Count on TEAM2 = TEAM1

where TEAM2 is null

union

select TEAM2, t2c

from team2Count

left join team1Count on TEAM2 = TEAM1

where TEAM1 is null

)

select p.FNAME, p.LNAME, pt.TEAM, pl.SHIRTNUM

from PLAYERTEAM pt

inner join PERSON P on pt.PID = P.PID

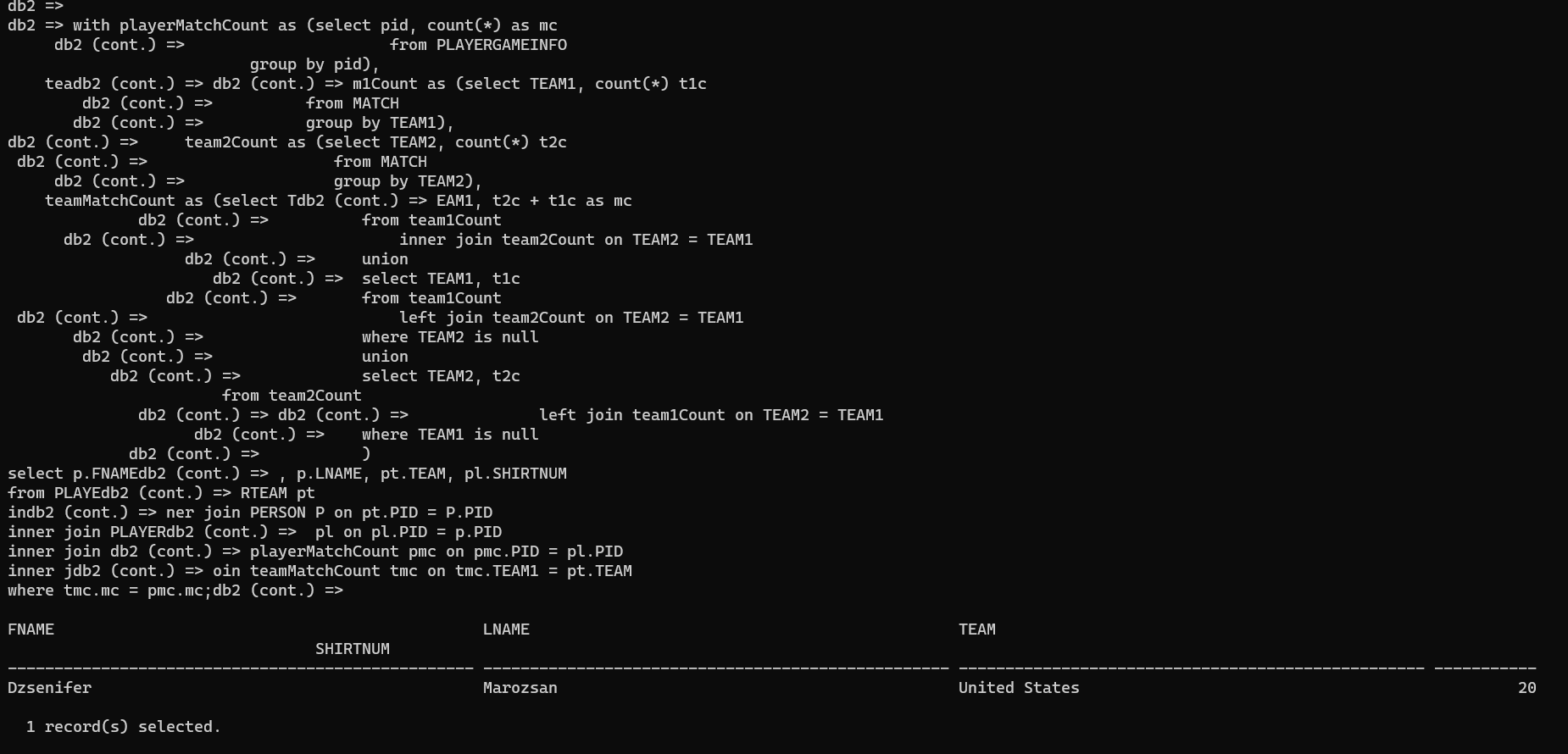
inner join PLAYER pl on pl.PID = p.PID

inner join playerMatchCount pmc on pmc.PID = pl.PID

inner join teamMatchCount tmc on tmc.TEAM1 = pt.TEAM

where tmc.mc = pmc.mc;

Table team1Count returns the number of matches a team in position team1 in the match table has participated in. Table team2Count returns the same thing but in position of team2. Table teamMatchCount returns the union of the 2 previous tables with the counts added when a team is in both positions team1 and team2 in the match table. The Table playerMatchCount returns the number of times a player has participated in matches. Then for the full query we selected the first name, last name from the person table, team from the team table, and shirt number form the player table. This is done by joining the Player team table with the person table on the player id then joining it to the player table on the player id. After that the resultant table was joined with playerMatchCount table on the player id then finally joining it to the teamMatchCount on the team value. Finally only the rows where the counts of the number of times the player has participate in games and the number of times the team has player, were selected.



1. SELECT COUNTRY,COALESCE(T1+T2,0) AS TOTALMATCH,COALESCE(TOTALGOAL,0) AS TOTALGOAL

FROM TEAM

LEFT OUTER JOIN (SELECT FORTEAM,COUNT(\*) AS TOTALGOAL

FROM GOALINFO

WHERE PENALTY = false

GROUP BY FORTEAM) AS GOAL ON GOAL.FORTEAM = TEAM.COUNTRY,

(SELECT TEAM1,COUNT(\*) AS T1

FROM MATCH

GROUP BY TEAM1),

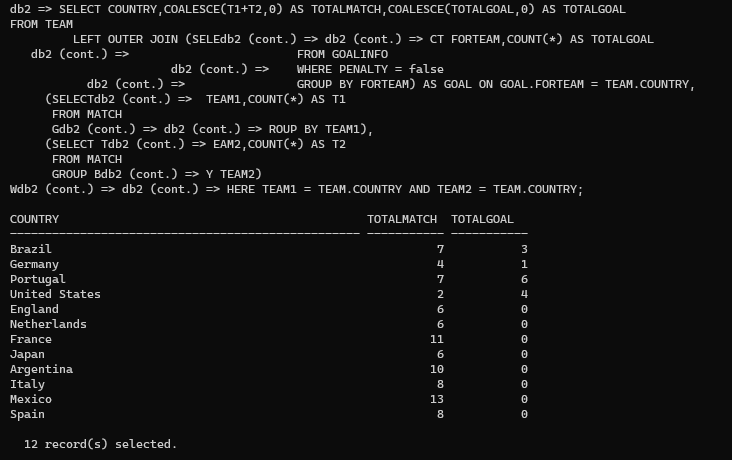
(SELECT TEAM2,COUNT(\*) AS T2

FROM MATCH

GROUP BY TEAM2)

WHERE TEAM1 = TEAM.COUNTRY AND TEAM2 = TEAM.COUNTRY;

For the match information we counted how many times a team occur as team1 or team2 and add the number up. For the goal information, we counted how many goal a team made not including penalties and then joined this information with the team table to get the result.



1. SELECT NAME AS STADIUMNAME, TEAM1, TEAM2, DATE,TICKETSSOLD,(MAXCAPACITY - TICKETSSOLD) AS TICKETSLEFT,AVGPRICESOLD, TOTALREVENUE

FROM STADIUM,MATCH,(SELECT TICKET.MID,COUNT(TICKET.TID) AS TICKETSSOLD,AVG(PRICE) AS AVGPRICESOLD, SUM(PRICE) AS TOTALREVENUE

FROM TICKET,MATCH,SALES

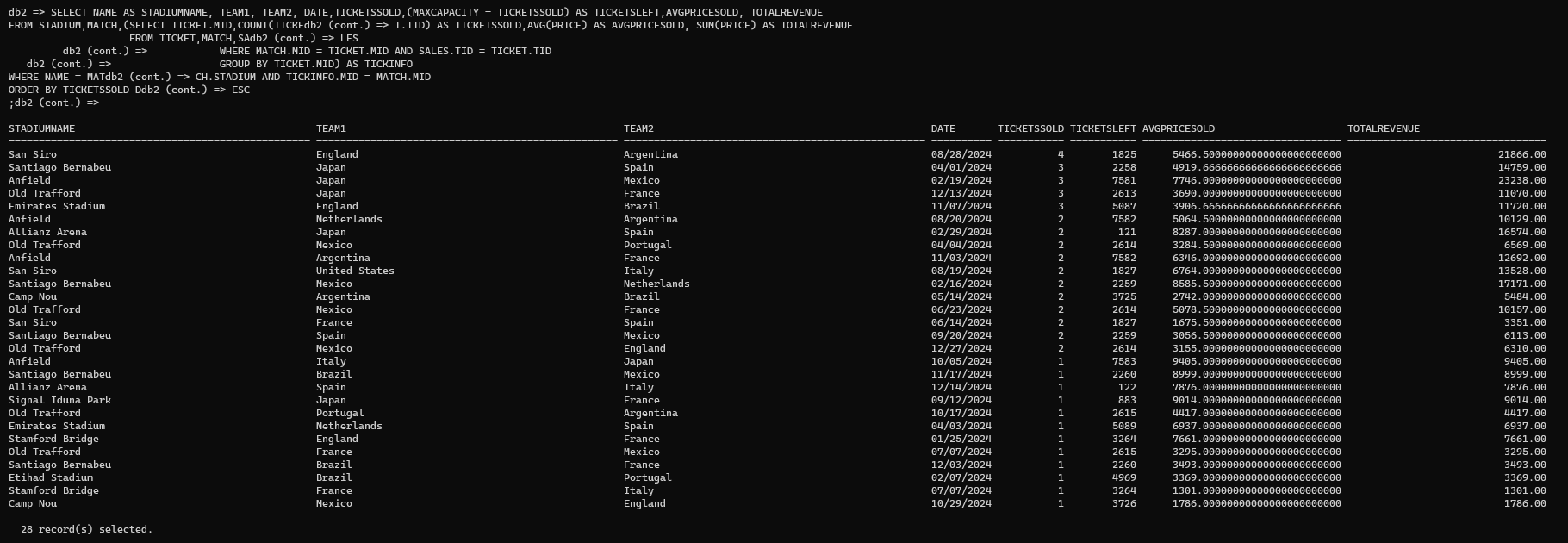
WHERE MATCH.MID = TICKET.MID AND SALES.TID = TICKET.TID

GROUP BY TICKET.MID) AS TICKINFO

WHERE NAME = MATCH.STADIUM AND TICKINFO.MID = MATCH.MID

ORDER BY TICKETSSOLD DESC;

For the ticket information we listed the stadium name,team1, team2, date, sold ticket , the left ticket and the average price of the tickets that are sold for every match with the total amount of revenue (price of all the tickets sold). First we selected the match id, the number of ticket id, and the average price from the ticket and match table grouped by the match id to get the sold ticket number, average price for each match, and the total revenue. Then using the above query as a subquery join with stadium and match table to get the stadium name, team 1, team 2, date of match, tickets sold, left seats using the maximum capacity minus the sold tickets, and finally the total revenue ordered by the number of tickets sold. This might give us insight into which matches were most popular and if location ie stadium had any effect on the number of tickets sold and many other things.



1. SELECT FNAME,LNAME,PERSON.PID,COALESCE(TOTALGOAL,0) AS TOTALGOAL,COALESCE(TOTALYELLOWCARD,0) AS TOTALYELLOWCARD,COALESCE(TOTALREDCARD,0) AS TOTALREDCARD

FROM PERSON

LEFT OUTER JOIN (SELECT PID,SUM(Y1 + Y2) AS TOTALYELLOWCARD,SUM(R) AS TOTALREDCARD

FROM PLAYERGAMEINFO

GROUP BY PID) AS CARD ON CARD.PID = PERSON.PID

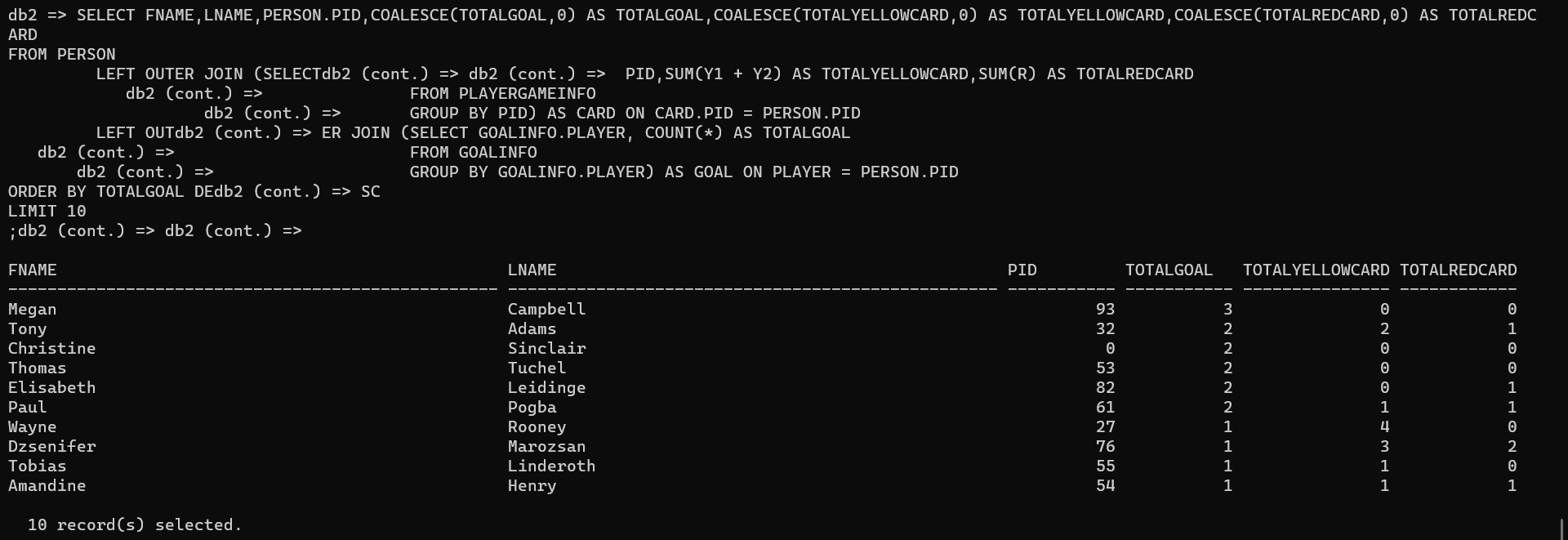
LEFT OUTER JOIN (SELECT GOALINFO.PLAYER, COUNT(\*) AS TOTALGOAL

FROM GOALINFO

GROUP BY GOALINFO.PLAYER) AS GOAL ON PLAYER = PERSON.PID

ORDER BY TOTALGOAL DESC;

We decided to make a player summary to see all player performance in this world cup. this query will show the first and last name of the player, player id, total goal they made, total number of yellow cards and red cards they received. The yellow card and red card information was obtained from the playerGameInfo table and the goal information from goalInfo table. After that, the person table was joined (left outer join) with these two query to get our desire information.



Player Information:

SQL ->

CREATE VIEW playerinfo AS

SELECT p.fname, p.lname, pl.shirtNum, p.DOB, t.country, t.officialName as association, t."group"

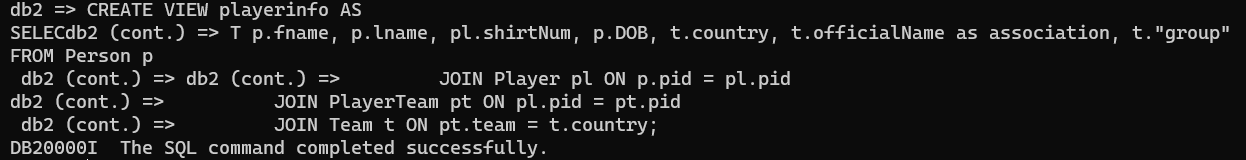
FROM Person p

JOIN Player pl ON p.pid = pl.pid

JOIN PlayerTeam pt ON pl.pid = pt.pid

JOIN Team t ON pt.team = t.country;

1. The create view returns the first name, last name from the person table, the shirt number of the player from the player table, then the date of birth from the person table, the country and the official name of the association and the group from the team table. This is done by joining the player table and the person table on the person id and then further joining it with the teamplayer table on the player id then finally joining the team table on the resultant table on the country of the team.



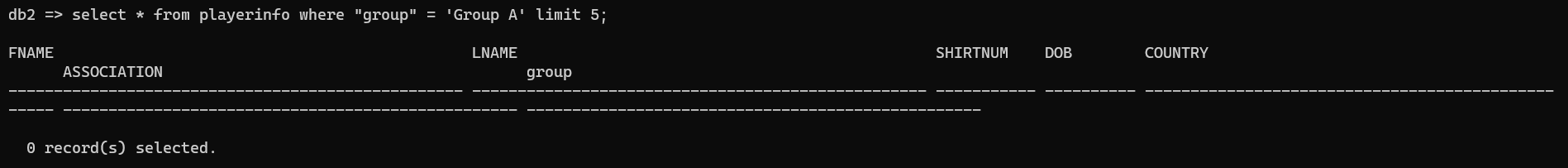




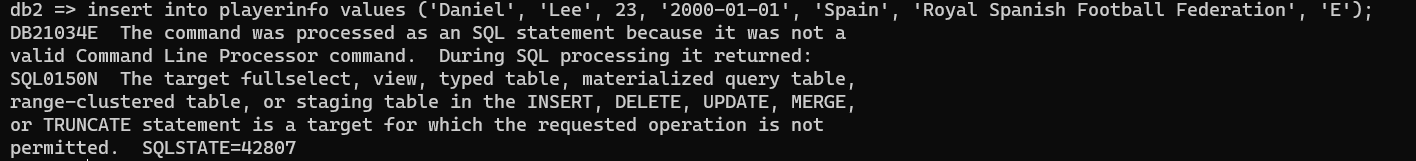
1. This is a picture where we replaced Group A with just A to account for the difference of the data Stored.



Here is another picture of the requested query







This happened because a view is a virtual table that is created on a base query on one or more base tables. The data is not stored directly but rather derived. This is why we can’t store values into the table. If we want to insert values it has to be through the base tables when then would get reflected in the view table.

Check Constraints:

The check constraint that we chose is to make the input of an invalid position of the player throw and error making this makes sure that only valid positions are entered. This screenshot shows both the creation of the table with the check constraint and the creation of a new record that fails.

