**Farcenal Football Club**

**Database Systems Design Project**

This document highlights the analysis, design and implementation of a database for Farcenal Football Club who are a professional football club in the English Premiere League.

Rahul Narayan

Database Systems

04/25/2014

**Table of Contents**

**Executive Summary**

**Entity Relationship Diagram**

**Tables: create statements, functional dependencies**

People table

Address table

EmployeeType table

Managers table

Players table

Coaches table

Doctors table

Countries table

workPermit table

financialSummary table

expenses table

income table

sponsorships table

sponsors table

sponsorType table

wages table

competitions table

internationalCompetitions table

scoutDetails table

region table

leagues table

positions table

scoutingHistory table

seasonObjectives table

injuryDetails table

specialty table

signonMedicals table

screening table

playerStats table

cupTied table

clubs table

**Views**

PlayersByPosition view

PlayersByCountry view

PlayersByInjury view

SponsorDetails view

WageDetails view

PlayersInInternationalTournaments view

EmployeeDetails view

**Reports & their queries**

GetInjuryListing

GetWorkPermitListing

GetSponsorDetails

TournamentParticipation

EmployeeDetails

**Stored Procedures**

GetInjuryDetails

GetWorkPermitDetails

GetTournamentParticipation

GetSponsorDetails

GetInjuryDetailsByPos

GetEmployeesByType

**Triggers Functions**

Check\_PrimaryAddress

Check\_isPlayer

Check\_isDoctor

**Security**

**Known Problems / Future enhancements**

**Executive Summary**

This document highlights the analysis and design of a database created for Farcenal Football Club. The database is meant to aid managers, and support staff in their day to day activities for the club. Some use cases include

Managers:

* Get injury updates
* Scouting information
* Assessment of available roster before an upcoming game
* Management of available roster before an upcoming game
* Look up goals established by management for a particular season

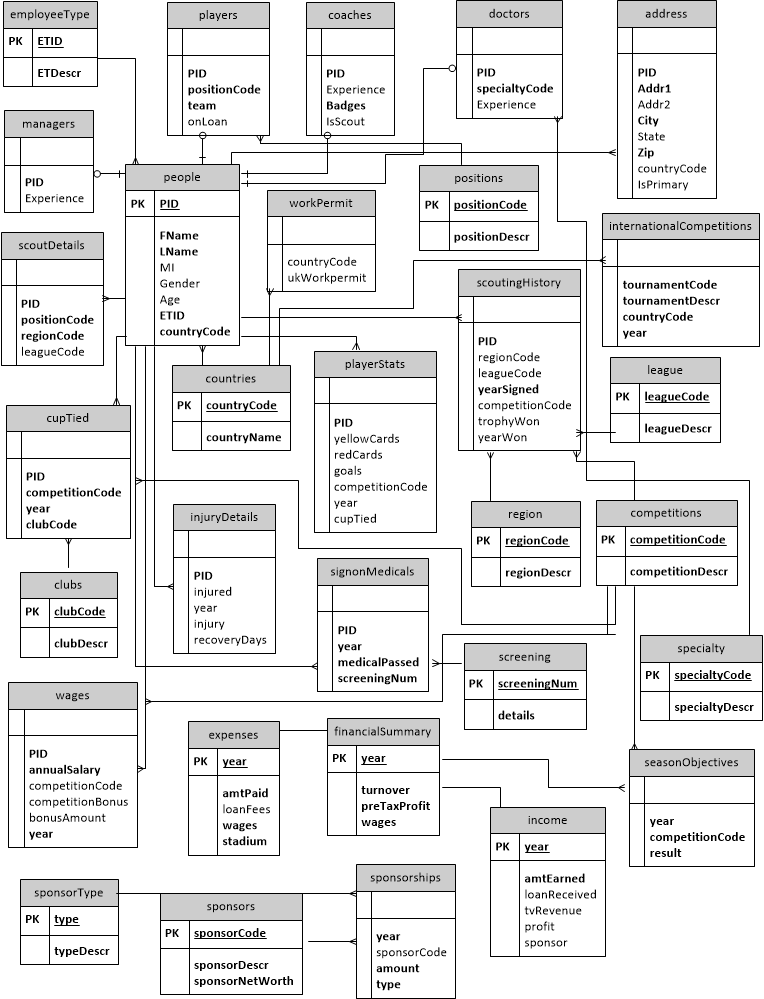
Doctors:

* Get health reports on current players in the squad
* Look up health history for current players in the squad to assess treatments

Financial employees:

* Get a snapshot of the club’s finances
* Financial details pertaining to sponsorship deals for any given year

**Entity Relationship Diagram**

****

**Tables**

**People table**

**Purpose:** Stores all personal and demographic information related to an employee of the club.

**Functional dependencies**

PID -> FName, LName, MI, Gender, Age, ETID, countryCode

**Table create statement**

CREATE TABLE people (

PID integer PRIMARY KEY,

FName varchar(100) NOT NULL,

LName varchar(100) NOT NULL,

MI varchar(1),

Gender varchar(1),

Age integer,

ETID integer NOT NULL,

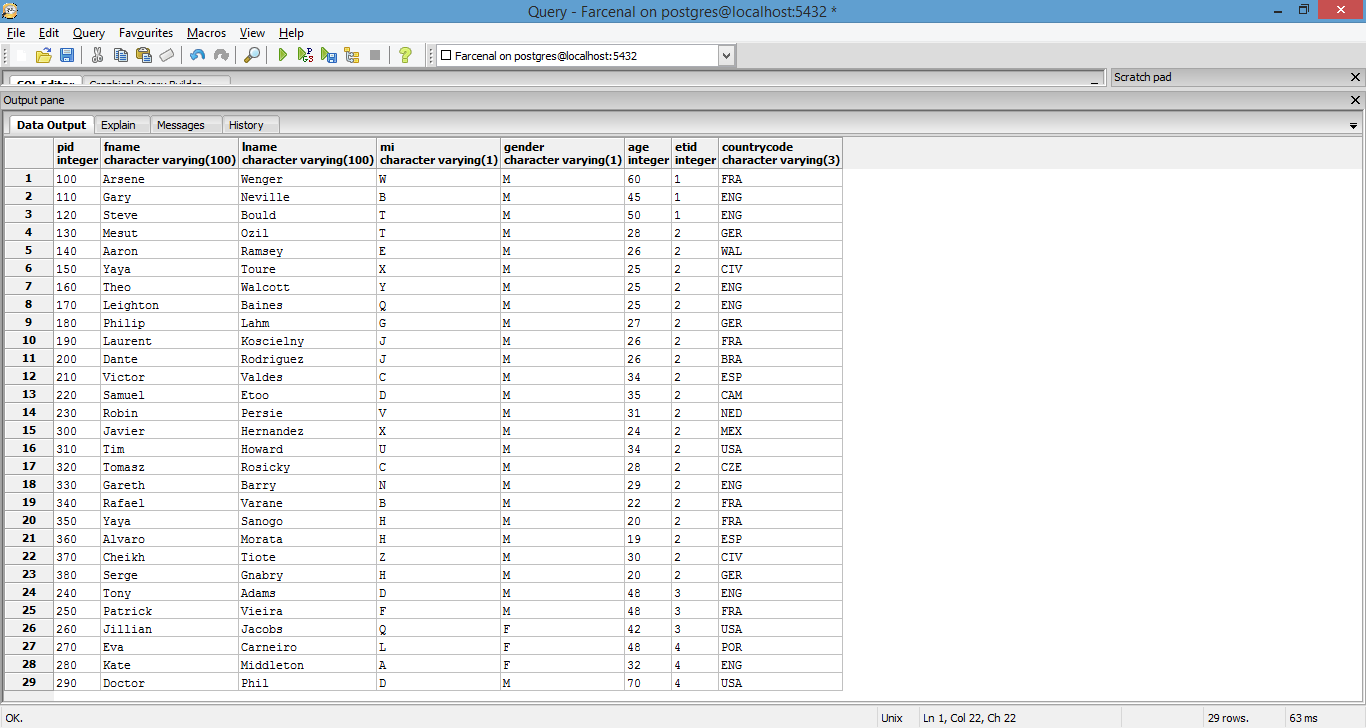
countryCode varchar(3) NOT NULL,

constraint gendercontraint CHECK (Gender = 'M' OR Gender = 'F'),

constraint etid CHECK (ETID = 1 OR ETID = 2 OR ETID = 3 OR ETID = 4)

);

**Sample data**



**Address table**

**Purpose:** For each PID established in the people table above, it stores address information.

**Functional dependencies**

PID -> Addr1, Addr2, City, State, Zip, countryCode, IsPrimary

**Table create statement**

CREATE TABLE address (

PID integer NOT NULL,

Addr1 varchar(100) NOT NULL,

Addr2 varchar(100),

City varchar(50) NOT NULL,

State varchar(2),

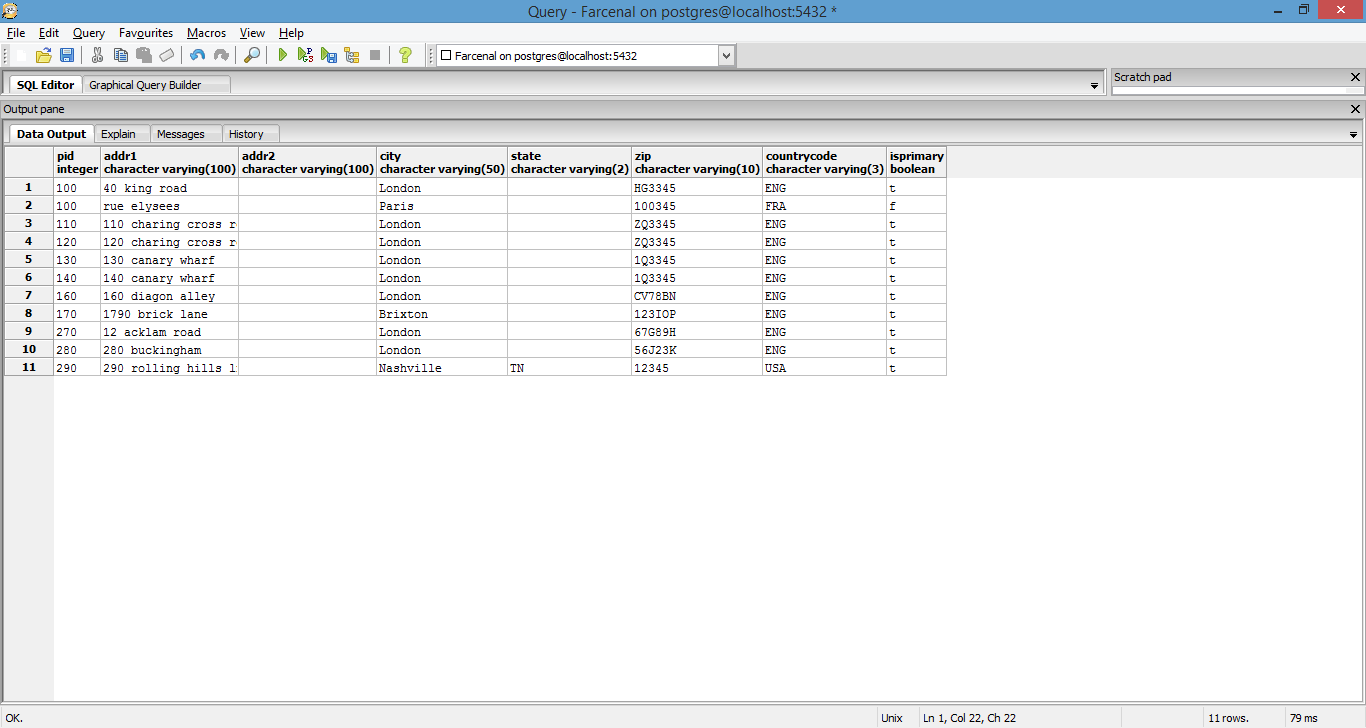
Zip varchar(10) NOT NULL,

countryCode varchar(3),

IsPrimary boolean

);

**Sample data**



**EmployeeType table**

**Purpose:** Establishes the types of employees that can be employed at the club.

**Functional dependencies**

ETID -> ETDescr

**Table create statement**

CREATE TABLE employeeType (

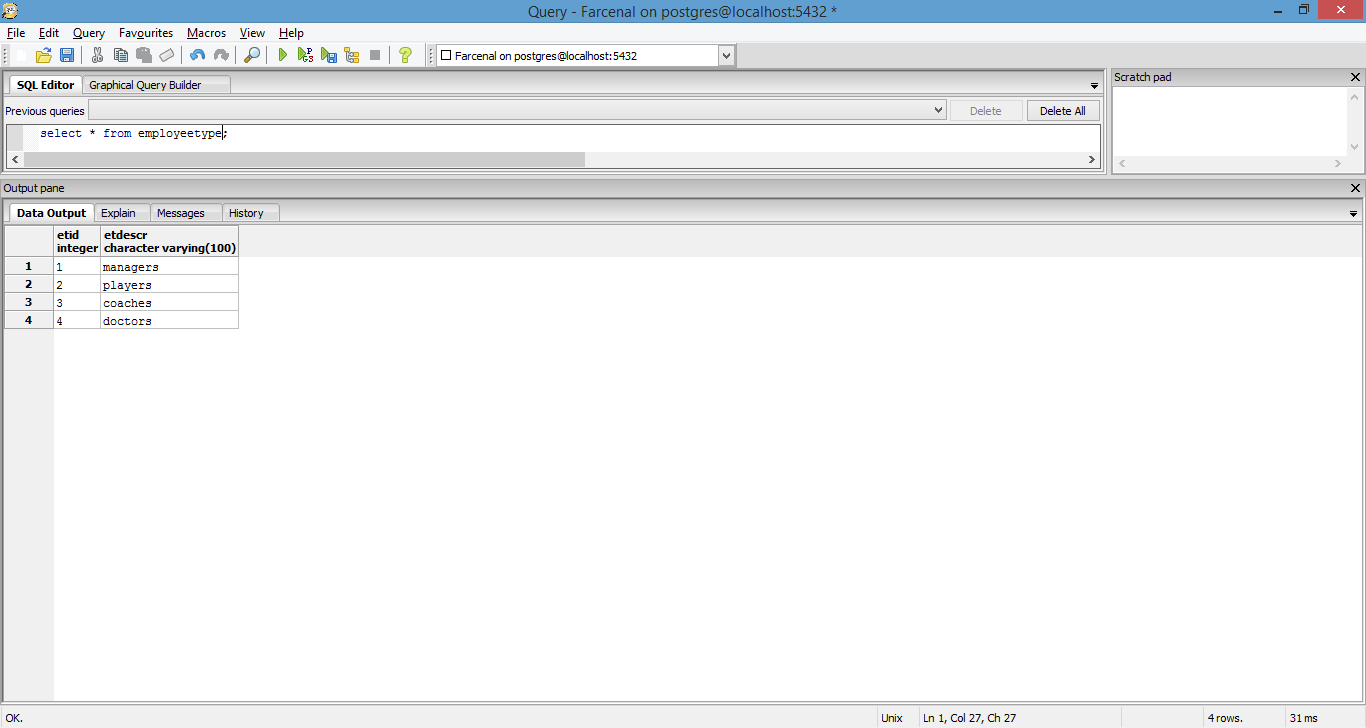
ETID integer PRIMARY KEY,

ETDescr varchar(100) NOT NULL,

constraint etid CHECK (ETID = 1 OR ETID = 2 OR ETID = 3 OR ETID = 4)

);

**Sample data**



**Managers table**

**Purpose:** To store information regarding the number of years of experience for all managers at the club.

**Functional dependencies**

PID -> Experience

**Table create statement**

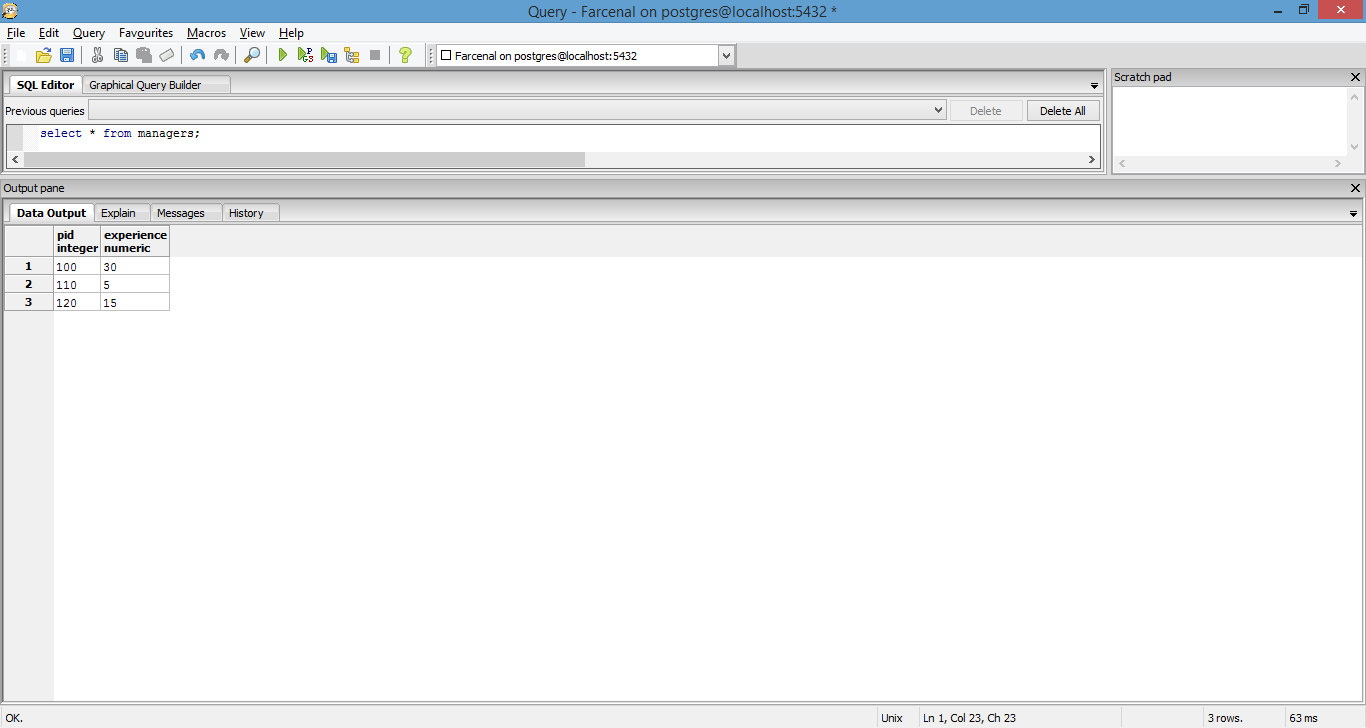
CREATE TABLE managers (

PID integer NOT NULL,

Experience decimal

);

**Sample data**

****

**Players table**

**Purpose:** To give summary information for each player who is on the roster.

**Functional dependencies**

PID -> positionCode, Team, onLoan,

**Table create statement**

CREATE TABLE players (

PID integer NOT NULL,

positionCode varchar(3) NOT NULL,

Team varchar(1) NOT NULL,

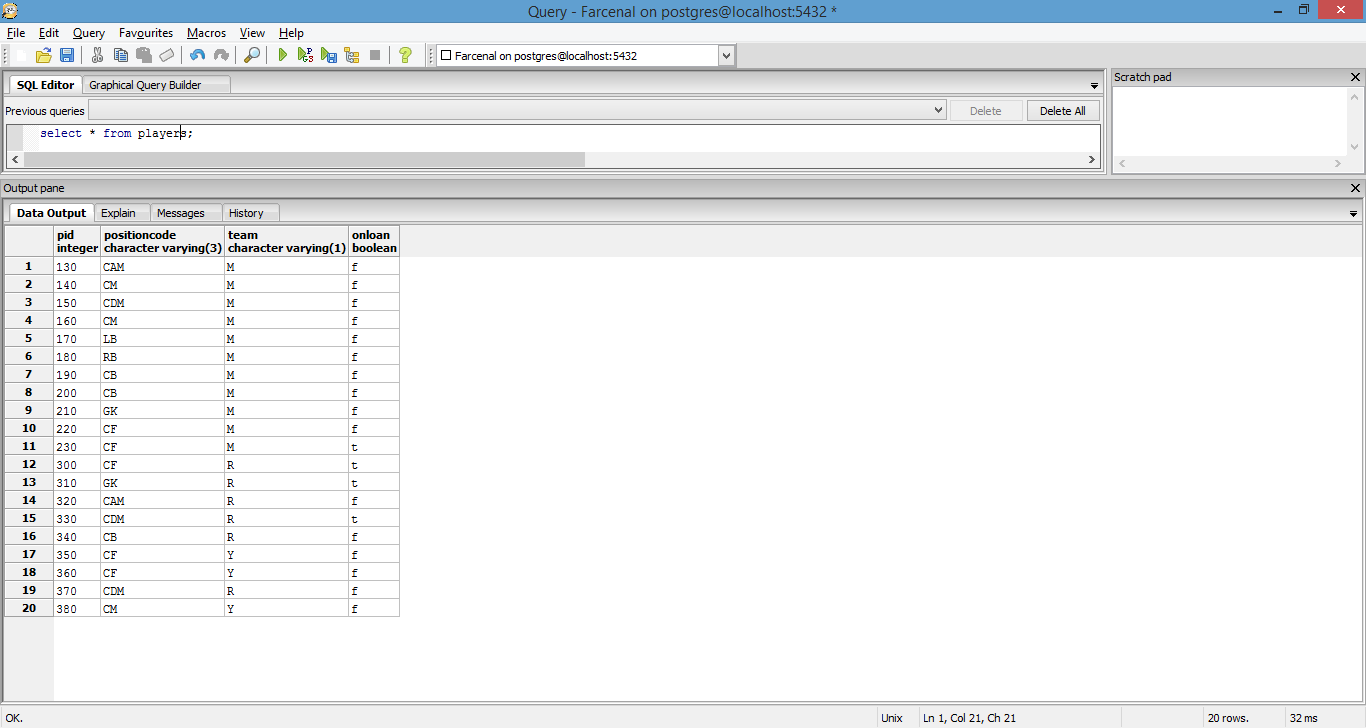
OnLoan boolean,

constraint positionCode CHECK (positionCode = 'CF' OR positionCode = 'RB' OR positionCode = 'LB' or positionCode = 'CB' or positionCode = 'CDM' or positionCode = 'CAM' or positionCode = 'CM' or positionCode = 'GK'),

constraint team CHECK (Team = 'M' OR Team = 'R' OR Team = 'Y')

);

**Sample data**



**Coaches table**

**Purpose:** To give summary information for each coach who is on staff.

**Functional dependencies**

PID -> Experience, Badges, IsScout

**Table create statement**

CREATE TABLE coaches (

PID integer NOT NULL,

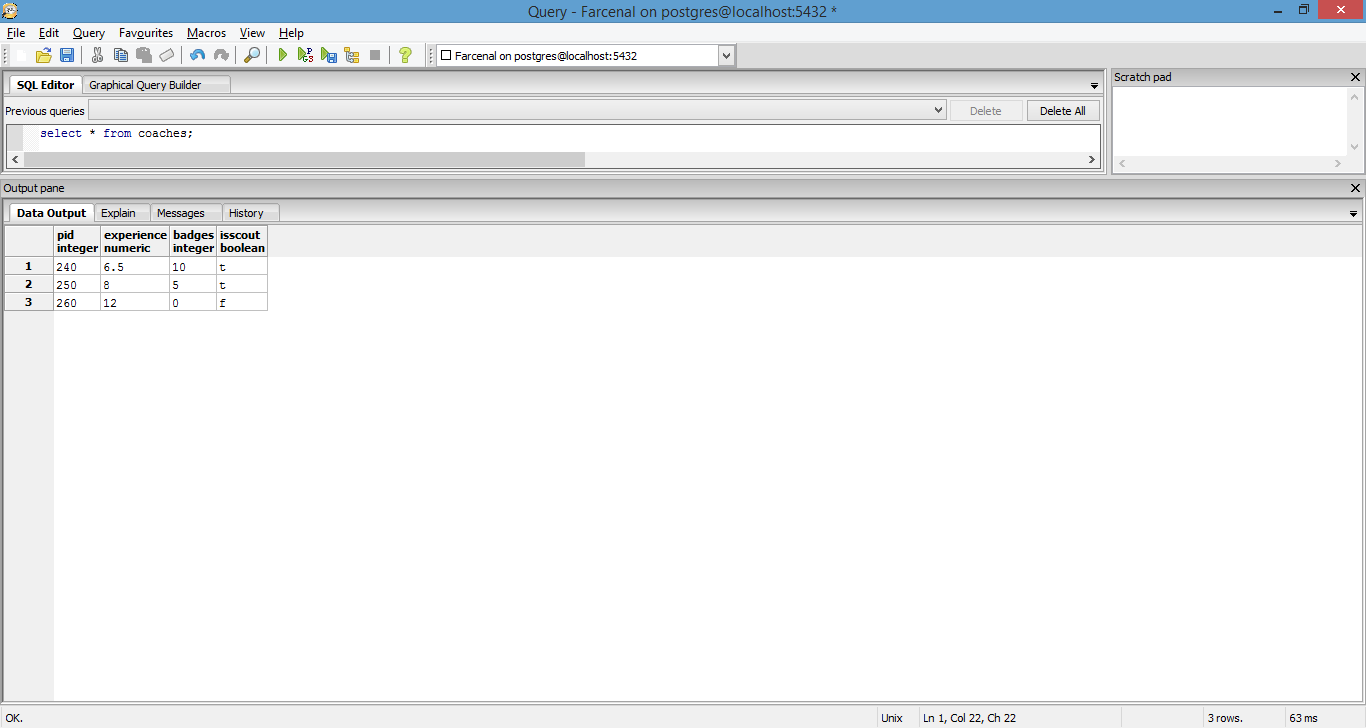
Experience decimal,

Badges integer NOT NULL,

IsScout boolean

);

**Sample data**



**Doctors table**

**Purpose:** To give information on the specialty and years of experience of each doctor on staff.

**Functional dependencies**

PID -> specialtyCode, Experience

**Table create statement**

CREATE TABLE doctors (

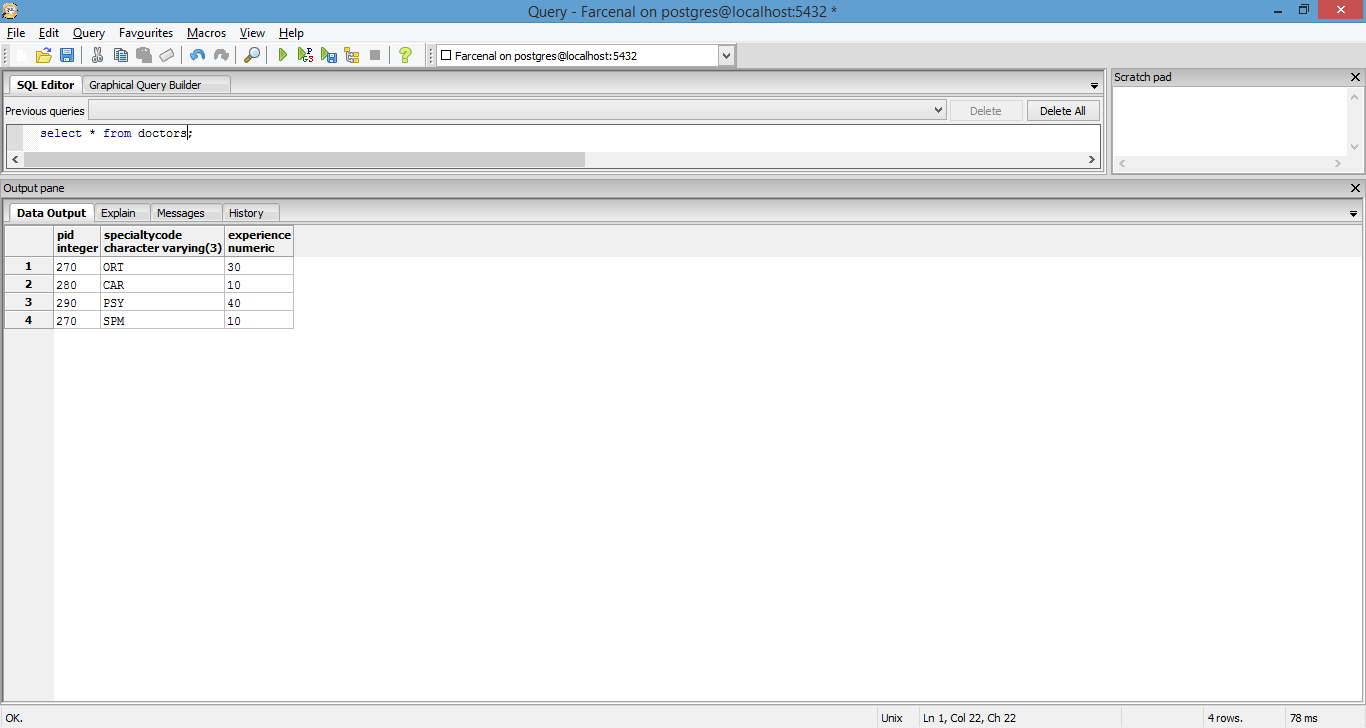
PID integer NOT NULL,

specialtyCode varchar(3) NOT NULL,

Experience decimal

);

**Sample data**



**Countries table**

**Purpose:** Due to the multi-national nature of staff, we need to maintain a table with details on nationality of each employee

**Functional dependencies**

countryCode -> countryName

**Table create statement**

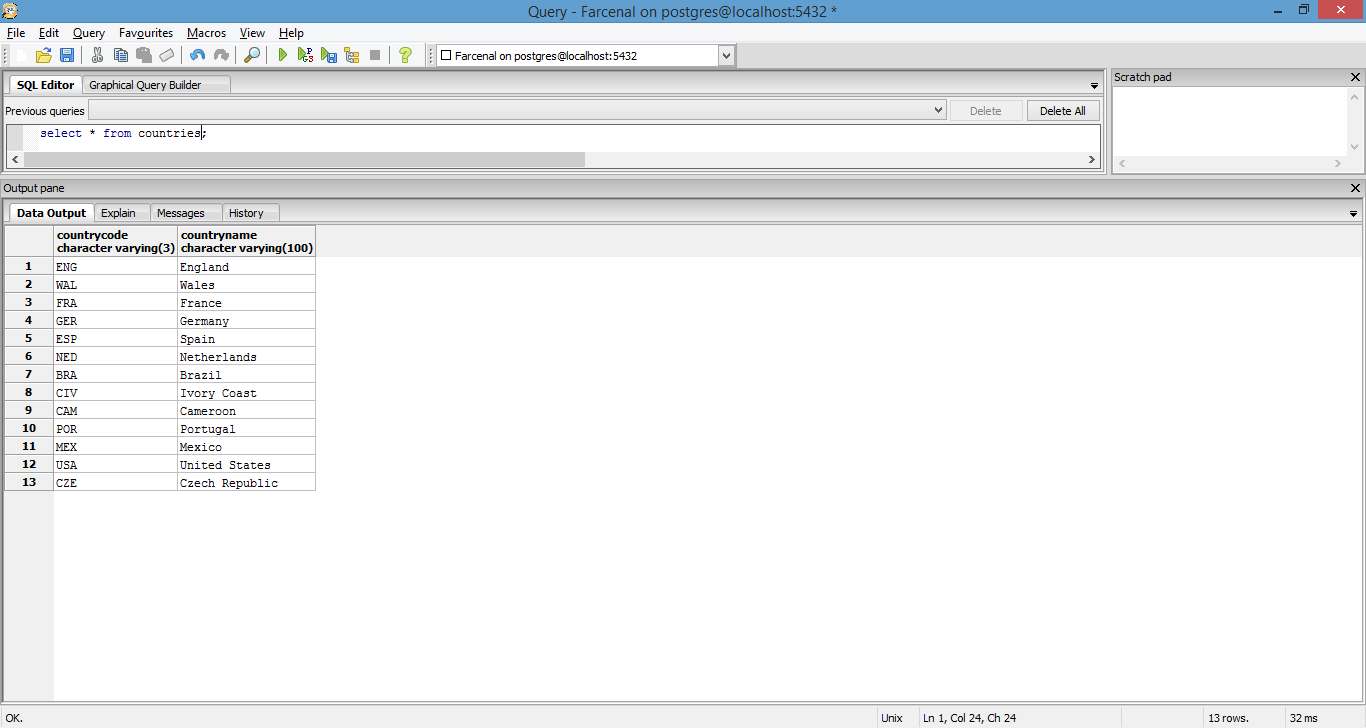
CREATE TABLE countries (

countryCode varchar(3) PRIMARY KEY,

countryName varchar(100) NOT NULL

);

**Sample data**



**WorkPermit table**

**Purpose:** Again, due to the multi-national nature of staff, we want to make sure they are authorized to work in the United Kingdom, where Farcenal FC is located.

**Functional dependencies**

countryCode -> ukWorkPermit

**Table create statement**

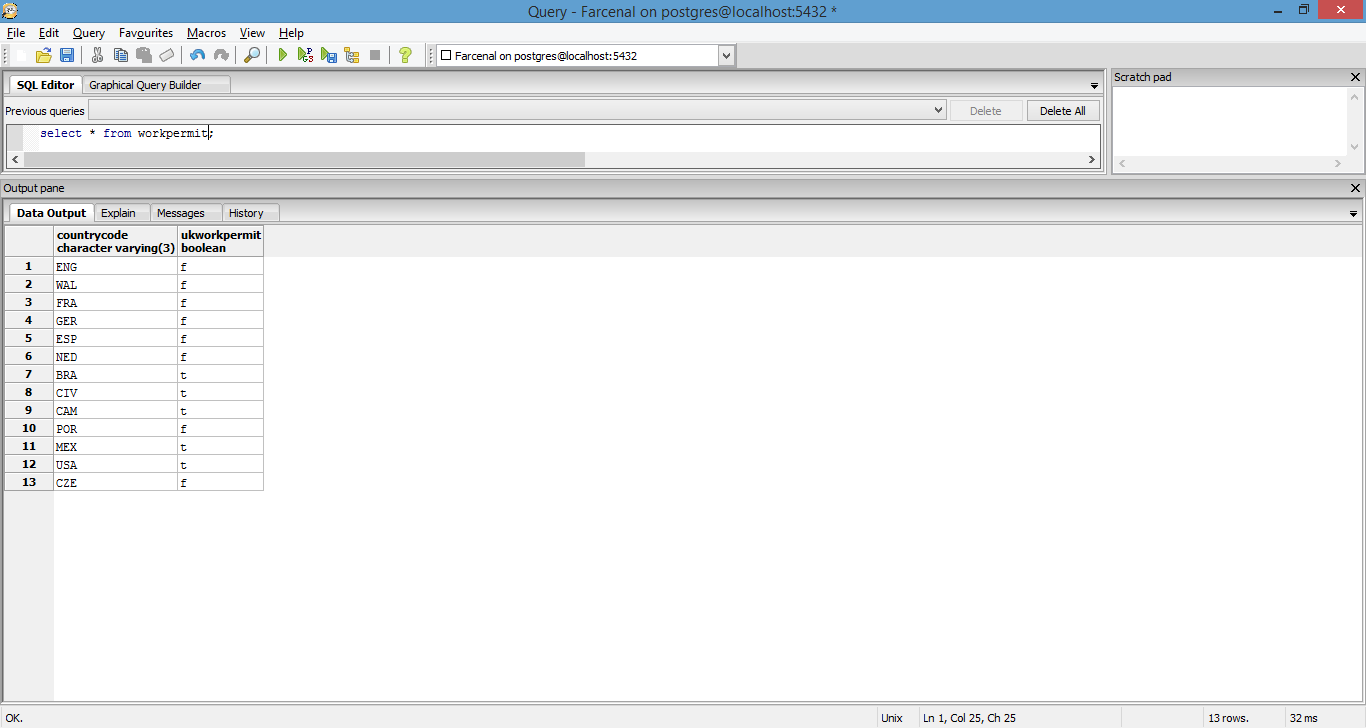
CREATE TABLE workPermit (

countryCode varchar(3),

ukWorkPermit boolean

);

**Sample data**



**financialSummary table**

**Purpose:** Gives snap shot information regarding club’s finances for a particular year.

**Functional dependencies**

Year -> turnover, preTaxProfit, wages

**Table create statement**

CREATE TABLE financialSummary (

year varchar(4) PRIMARY KEY,

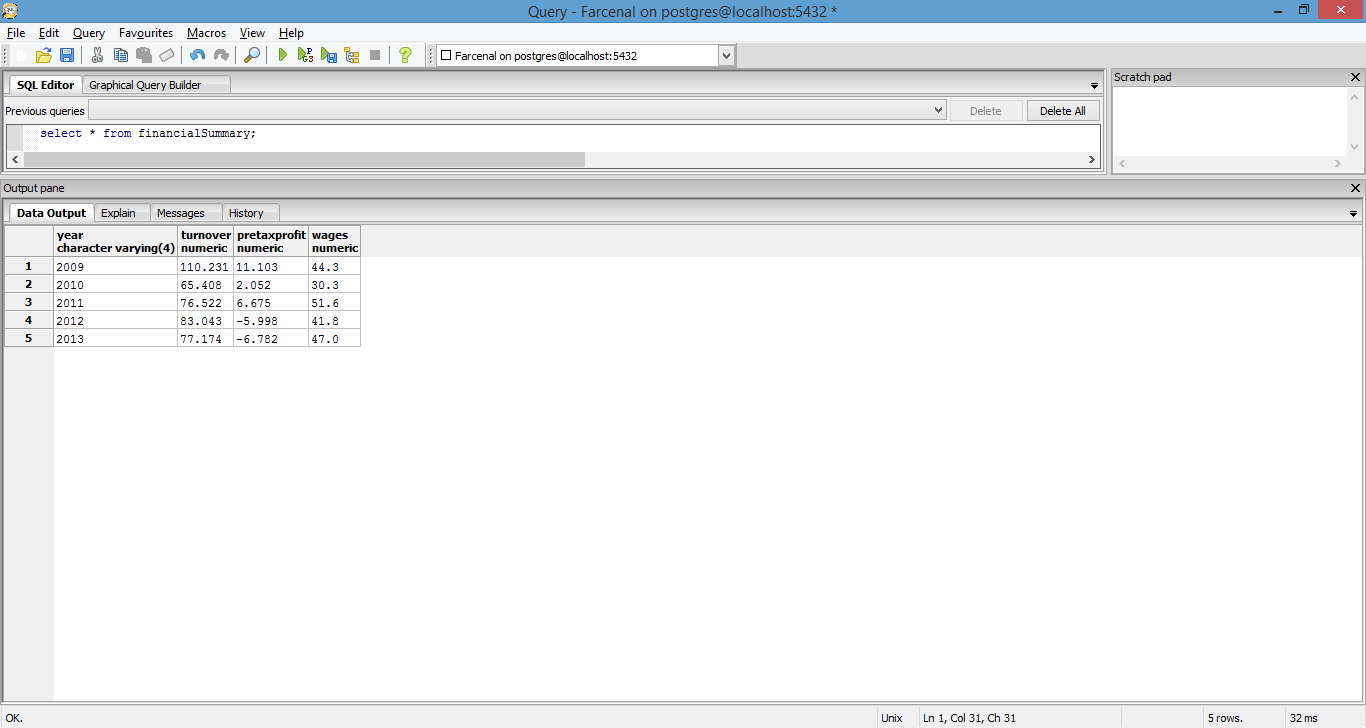
turnover decimal NOT NULL,

preTaxProfit decimal NOT NULL,

wages decimal NOT NULL

);

**Sample data**



**Expenses table**

**Purpose:** Gives breakdown of club expenses by category for a particular year.

**Functional dependencies**

Year -> amtPaid, loanFees, wages, stadium

**Table create statement**

CREATE TABLE expenses (

year varchar(4) PRIMARY KEY,

amtPaid decimal NOT NULL,

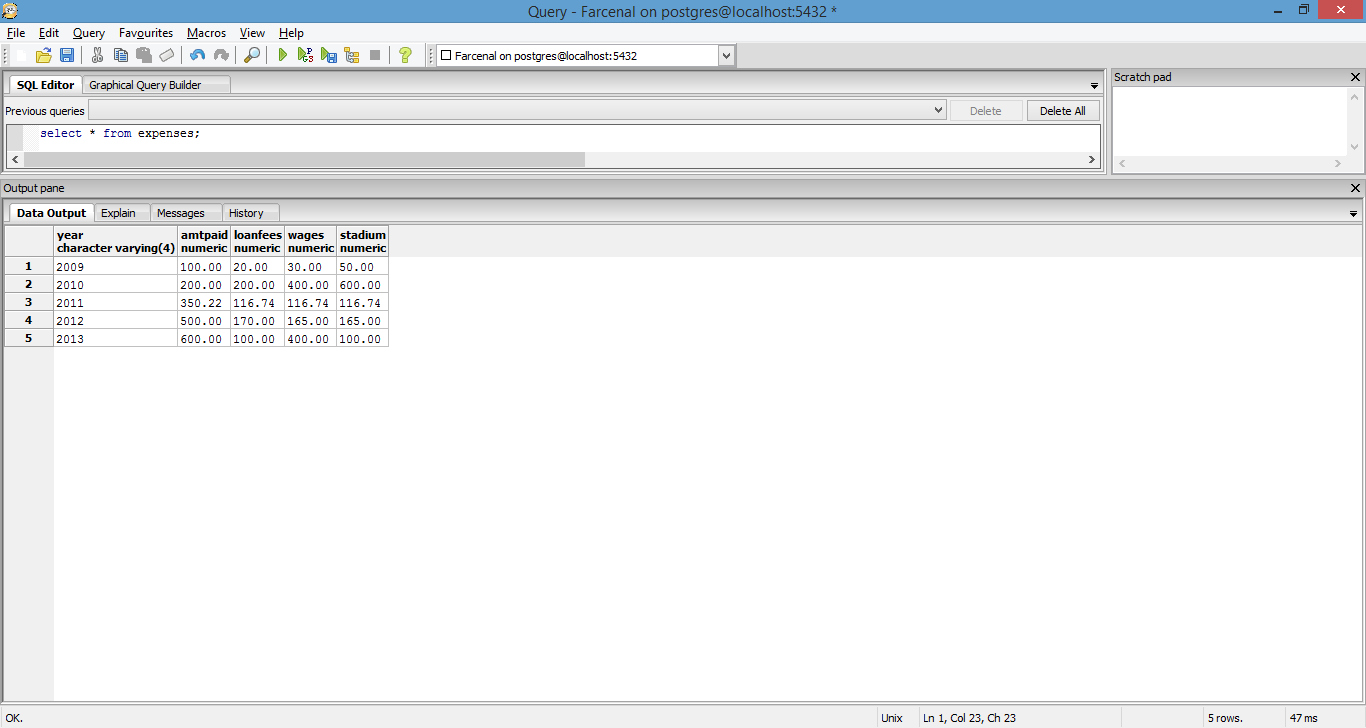
loanFees decimal,

wages decimal NOT NULL,

stadium decimal NOT NULL

);

**Sample data**



**Income table**

**Purpose:** Gives breakdown of club income by category for a particular year.

**Functional dependencies**

Year -> amtEarned, loanReceived, tvRevenue, profit, sponsor

**Table create statement**

CREATE TABLE income (

year varchar(4) PRIMARY KEY,

amtEarned decimal NOT NULL,

loanReceived decimal,

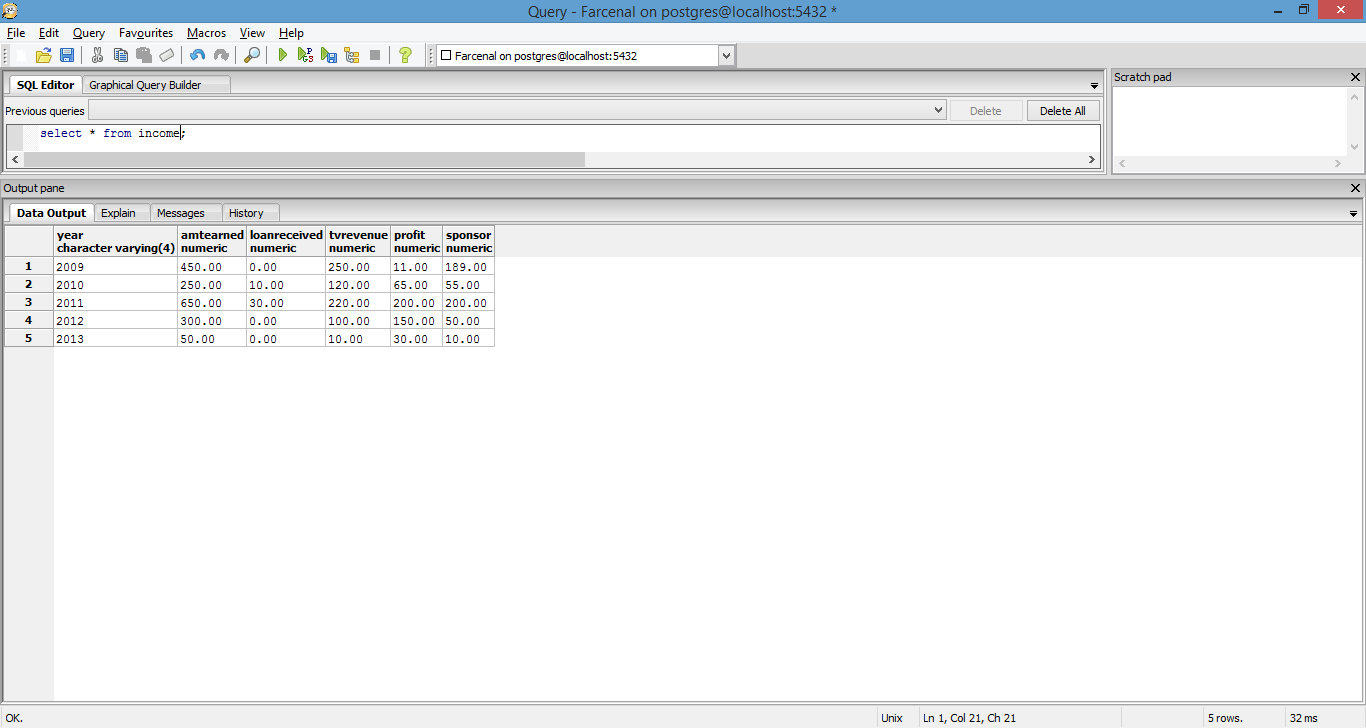
tvRevenue decimal,

profit decimal,

sponsor decimal

);

**Sample data**



**Sponsorships table**

**Purpose:** Gives base level sponsor information for each sponsor associated with the club in a given year.

**Functional dependencies**

SponsorCode -> year, amount, type

**Table create statement**

CREATE TABLE sponsorships (

year varchar(4),

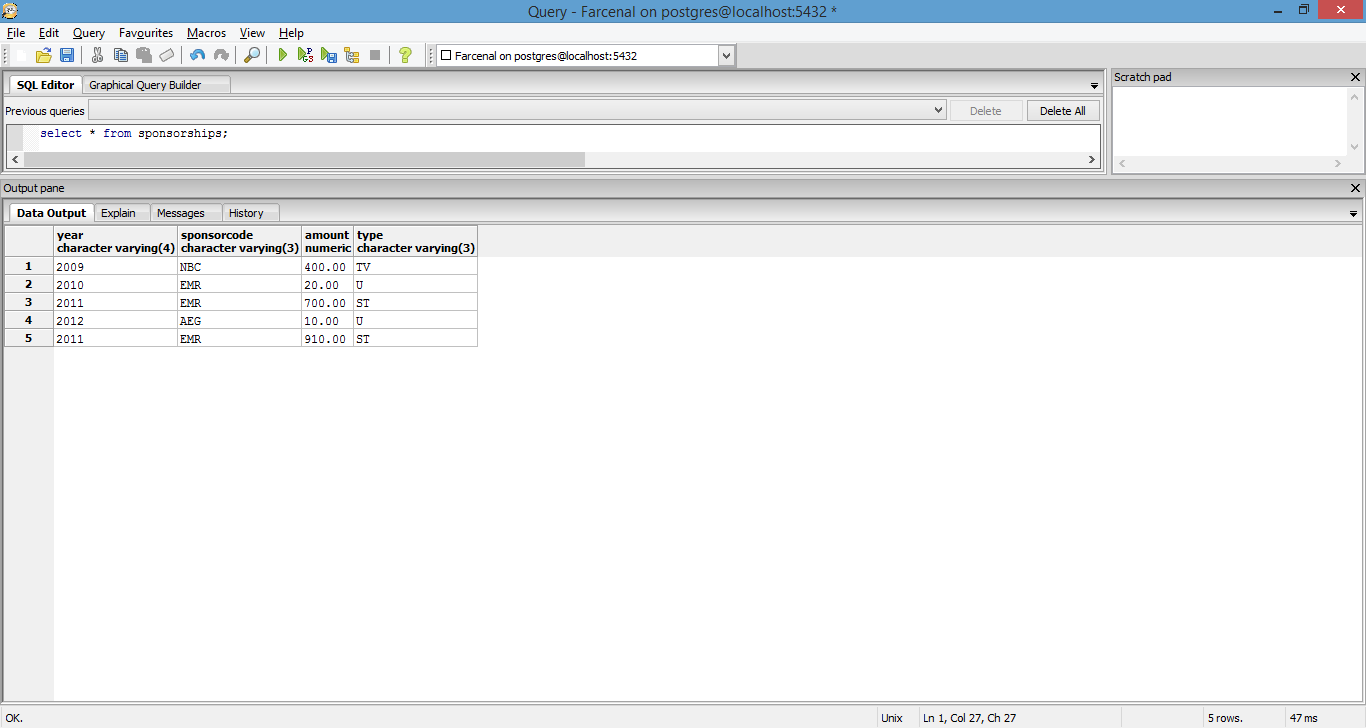
sponsorCode varchar(3),

amount decimal NOT NULL,

type varchar(3)

);

**Sample data**



**Sponsors table**

**Purpose:** Personal details about each sponsor featured in the table above.

**Functional dependencies**

SponsorCode -> sponsorDescr, sponsorNetWorth

**Table create statement**

CREATE TABLE sponsors (

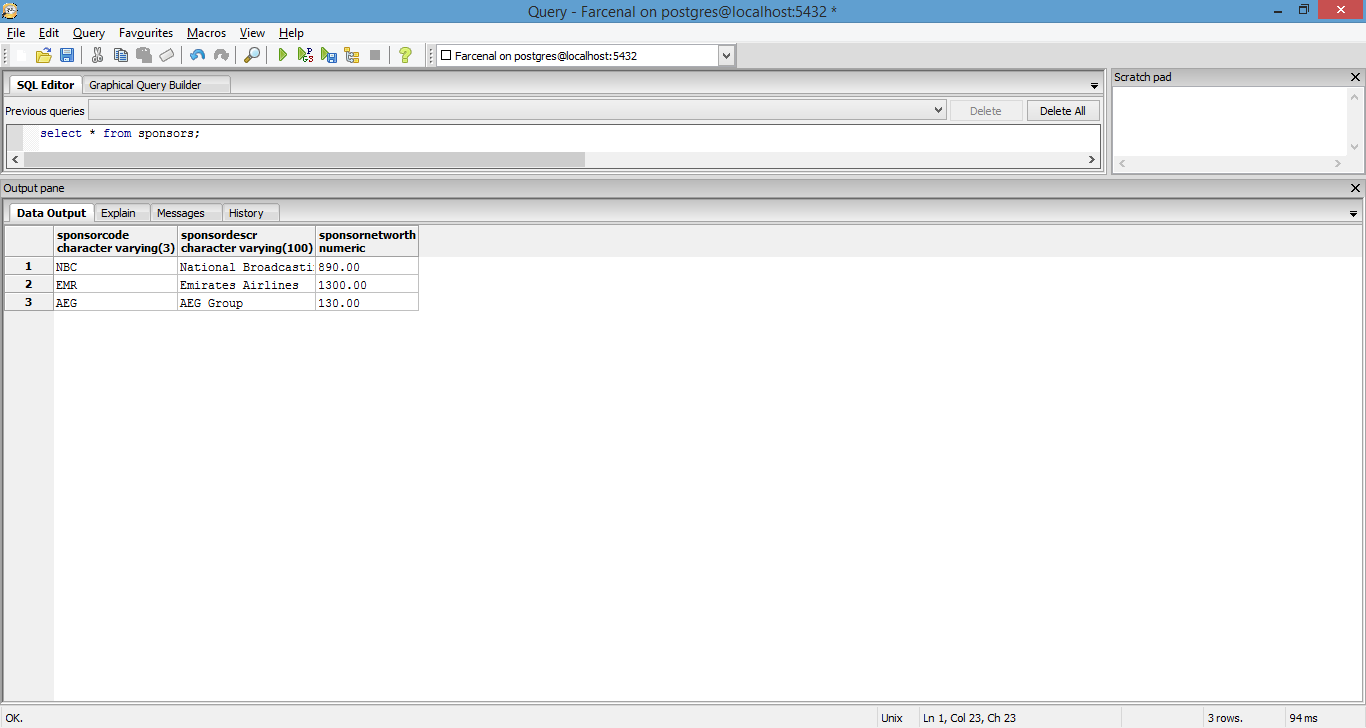
sponsorCode varchar(3) PRIMARY KEY,

sponsorDescr varchar(100) NOT NULL,

sponsorNetWorth decimal NOT NULL

);

**Sample data**



**sponsorType table**

**Purpose:** Information about the type of sponsorship for each sponsor associated with the club. Example: TV sponsor, versus stadium naming rights etc

**Functional dependencies**

Type -> typeDescr

**Table create statement**

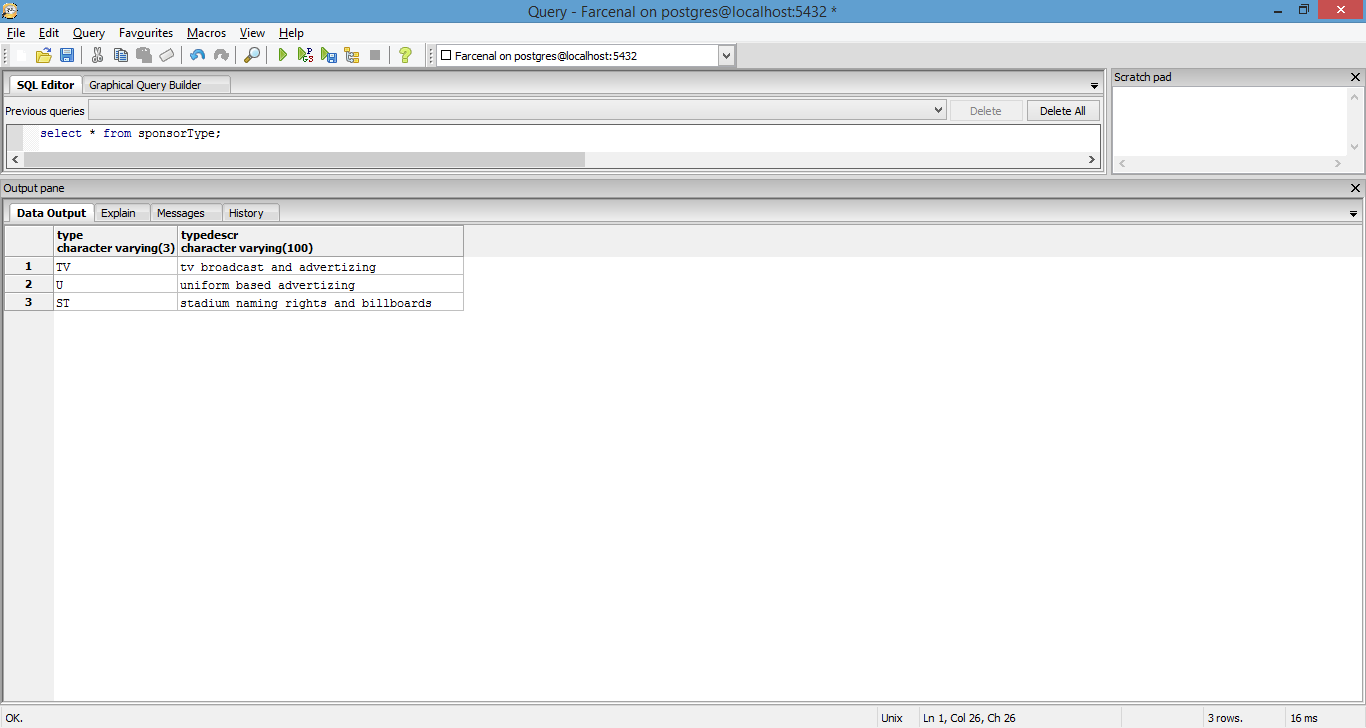
CREATE TABLE sponsorType (

type varchar(3) PRIMARY KEY,

typeDescr varchar(100) NOT NULL

);

**Sample data**



**Wages table**

**Purpose:** breakdown of a particular employee’s wage into sub-components making up the full wages for a particular year. Will also let the viewer determine whether an employee is being given performance based bonuses for a year and if so, the performances that this is based on.

**Functional dependencies**

PID -> annualSalary, competitionCode, competitionBonus, bonusAmount

**Table create statement**

CREATE TABLE wages (

PID integer NOT NULL,

annualSalary decimal NOT NULL,

competitionCode varchar(3),

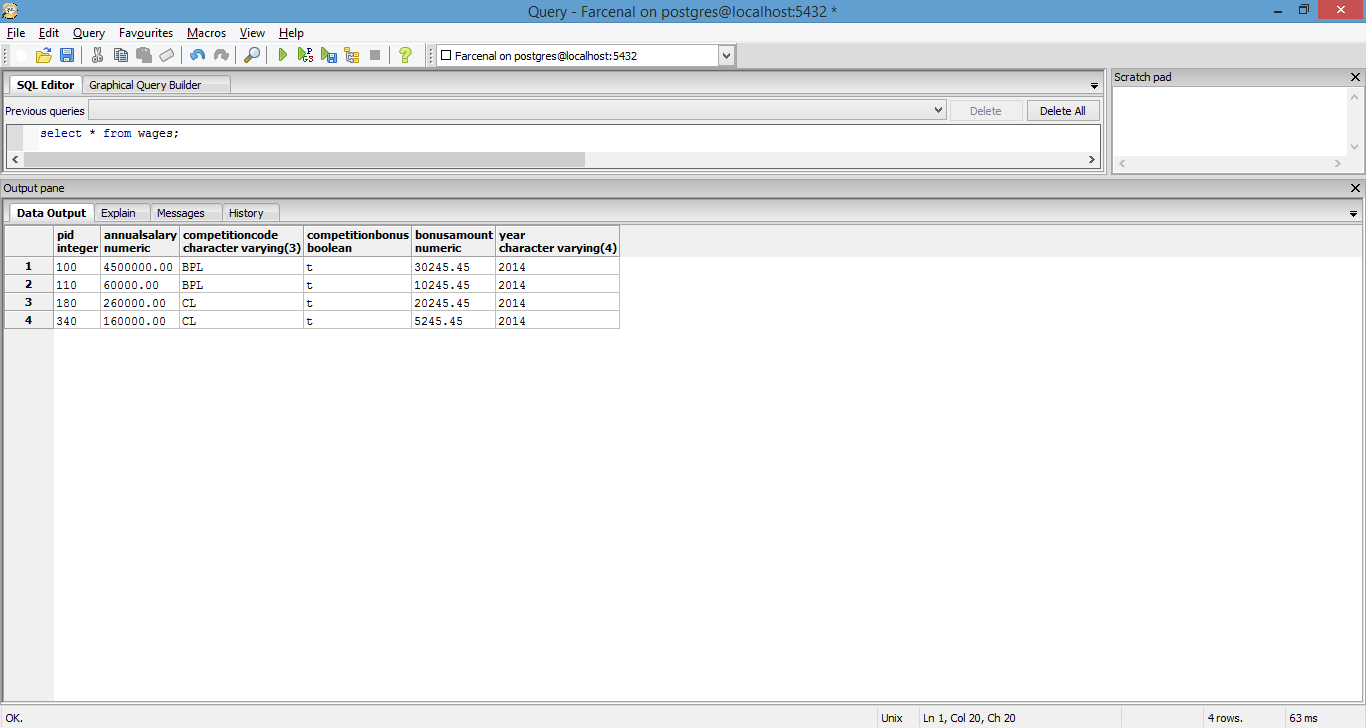
competitionBonus boolean,

bonusAmount decimal,

year varchar(4) NOT NULL

);

**Sample data**



**Competitions table**

**Purpose:** Details about each club based competition the club has entered.

**Functional dependencies**

competitionCode -> competitionDescr

**Table create statement**

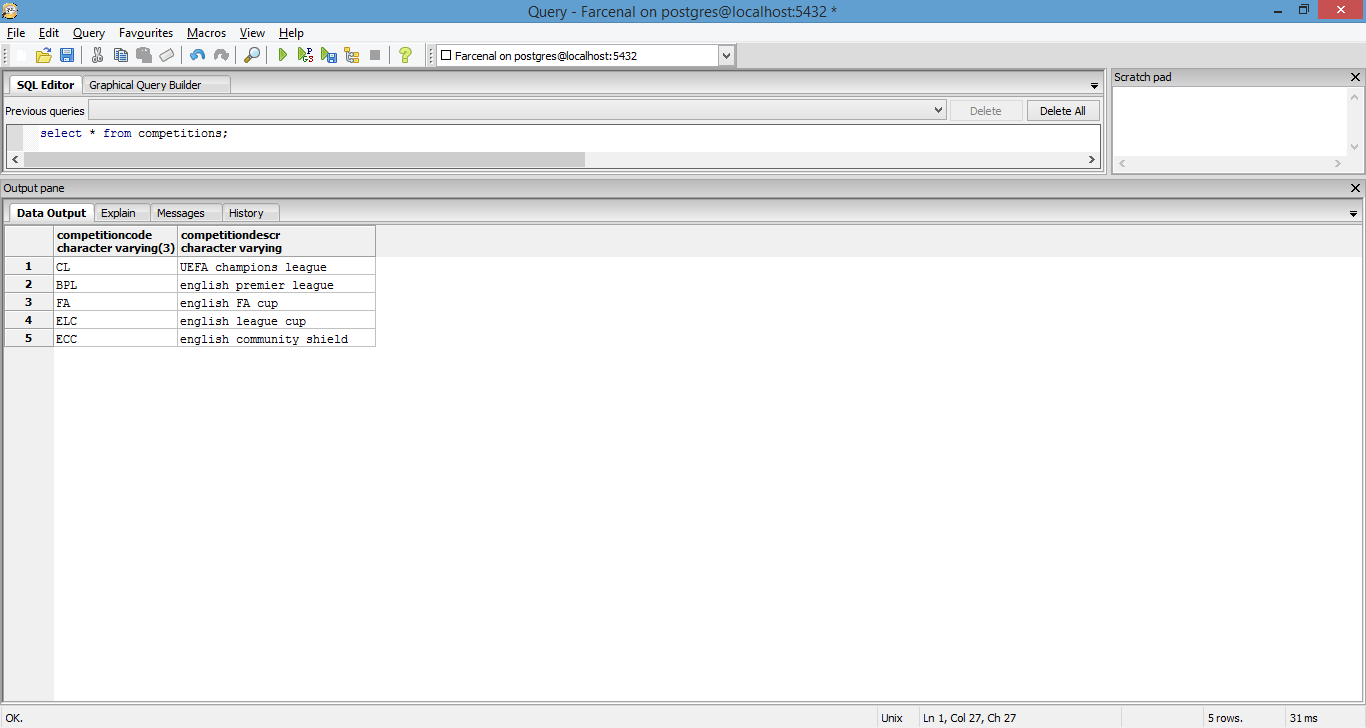
CREATE TABLE competitions (

competitionCode varchar(3) PRIMARY KEY,

competitionDescr varchar NOT NULL

);

**Sample data**



**internationalCompetitions table**

**Purpose:** Details about non club based i.e. international competitions that the club’s players may be participating in.

**Functional dependencies**

tournamentCode, year -> tournamentDesr, countryCode

**Table create statement**

CREATE TABLE internationalCompetitions (

tournamentCode varchar(3) NOT NULL,

tournamentDescr varchar(100) NOT NULL,

countryCode varchar(3) NOT NULL,

year varchar(4) NOT NULL

);

**scoutDetails table**

**Purpose:** What position(s) is a scout looking to recruit for, and where is he/she doing their recruitment ?

**Functional dependencies**

PID -> positionCode, regionCode, leagueCode

**Table create statement**

CREATE TABLE scoutDetails (

PID integer NOT NULL,

positionCode varchar(3) NOT NULL,

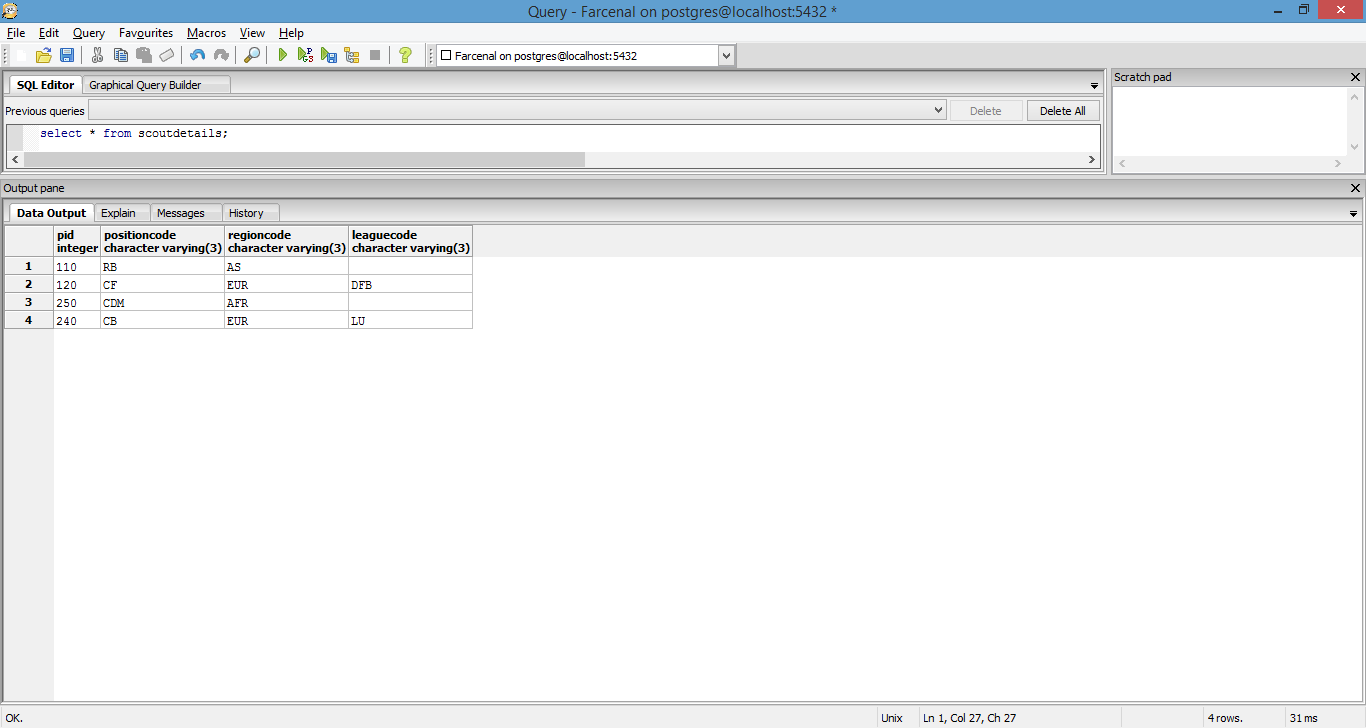
regionCode varchar(3) NOT NULL,

leagueCode varchar(3),

constraint positionCode CHECK (positionCode = 'CF' OR positionCode = 'RB' OR positionCode = 'LB' or positionCode = 'CB' or positionCode = 'CDM' or positionCode = 'CAM' or positionCode = 'CM')

);

**Sample data**



**Region table**

**Purpose:** Detailed information on the regions that a scout may recruit from

**Functional dependencies**

regionCode -> regionDescr

**Table create statement**

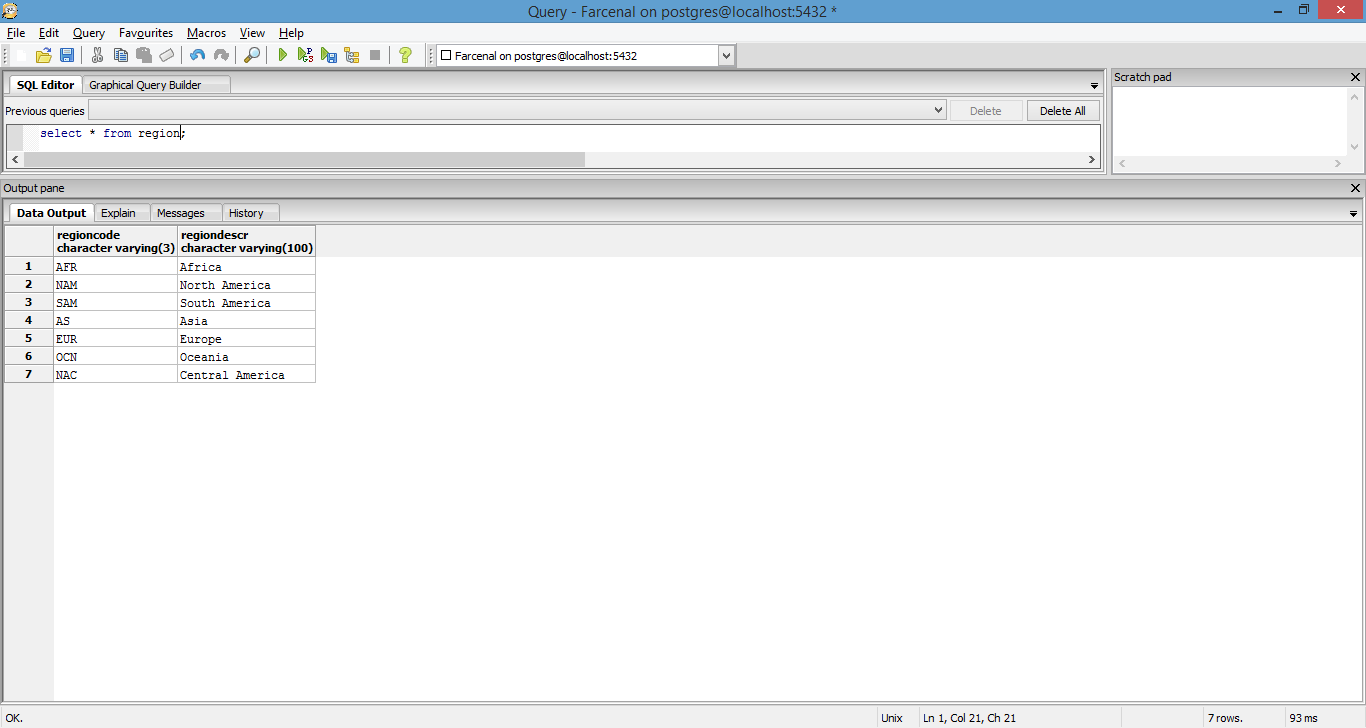
CREATE TABLE region (

regionCode varchar(3) PRIMARY KEY,

regionDescr varchar(100) NOT NULL

);

**Sample data**



**Leagues table**

**Purpose:** Detailed information about the leagues that the club plays in/a scout may recruit from.

**Functional dependencies**

leagueCode -> leagueDescr

**Table create statement**

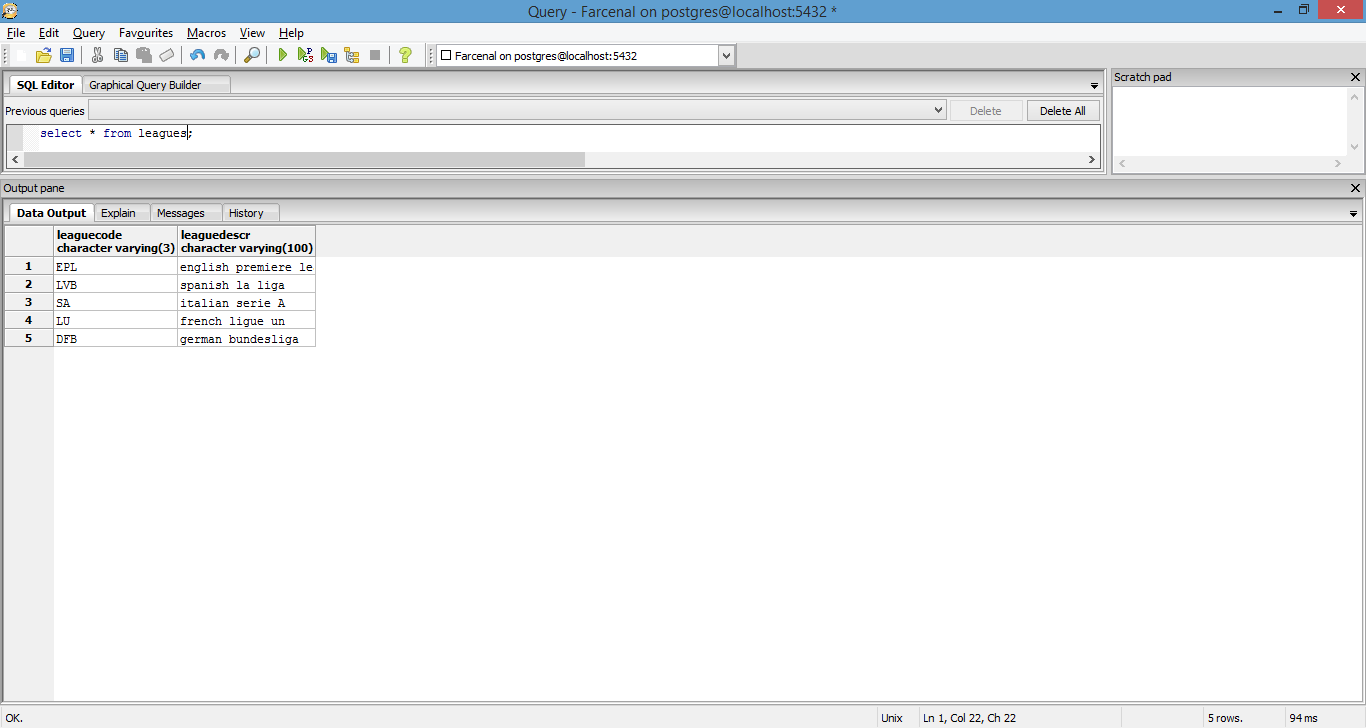
CREATE TABLE leagues (

leagueCode varchar(3) PRIMARY KEY,

leagueDescr varchar(100) NOT NULL

);

**Sample data**



**Positions table**

**Purpose:** Description of each of the positions a player at the club would play under. Can also be used to perform queries while scouting. **Example:** I need 3 strikers because all of my first-team strikers are injured.

**Functional dependencies**

positionCode -> positionDescr

**Table create statement**

CREATE TABLE positions (

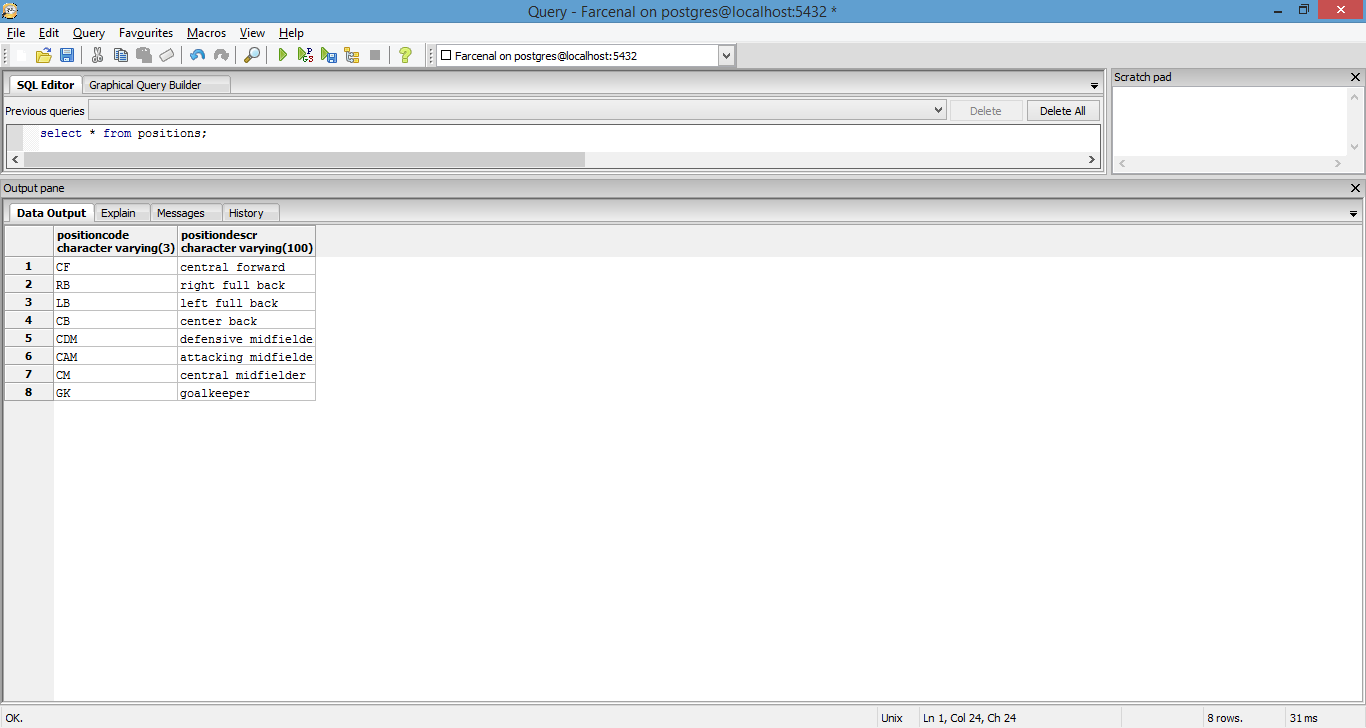
positionCode varchar(3) PRIMARY KEY,

positionDescr varchar(100) NOT NULL,

constraint positionCode CHECK (positionCode = 'CF' OR positionCode = 'RB' OR positionCode = 'LB' or positionCode = 'CB' or positionCode = 'CDM' or positionCode = 'CAM' or positionCode = 'CM' or positionCode = 'GK')

);

**Sample data**

****

**scoutingHistory table**

**Purpose:** To evaluate scouting successes or failures. **Example:** How many players did we scout this year and out of those how many worked out and won trophies for the club? Of those that won trophies, where did we recruit them from?

**Functional dependencies**

PID, regionCode, leagueCode, yearSigned, competitionCode, trophyWon, yearWon ->

**Table create statement**

CREATE TABLE scoutingHistory (

PID integer NOT NULL,

regionCode varchar(3),

leagueCode varchar(3),

yearSigned varchar(4) NOT NULL,

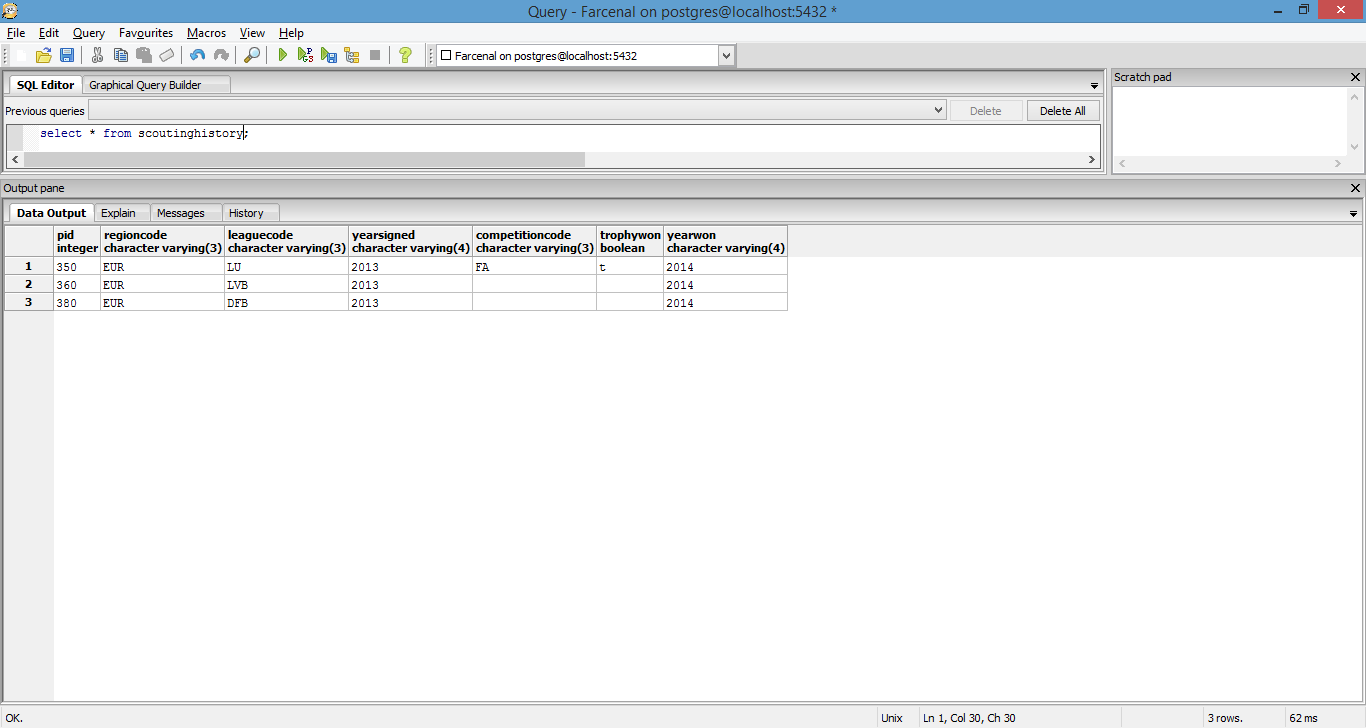
competitionCode varchar(3),

trophyWon boolean,

yearWon varchar(4)

);

**Sample data**



**seasonObjectives table**

**Purpose:** For a given season, what are our aims? Can be used to evaluate a manager’s performance or determine whether he/she should be given a performance based bonus.

**Functional dependencies**

Year -> competitionCode, result

**Table create statement**

CREATE TABLE seasonObjectives (

year varchar(4) NOT NULL,

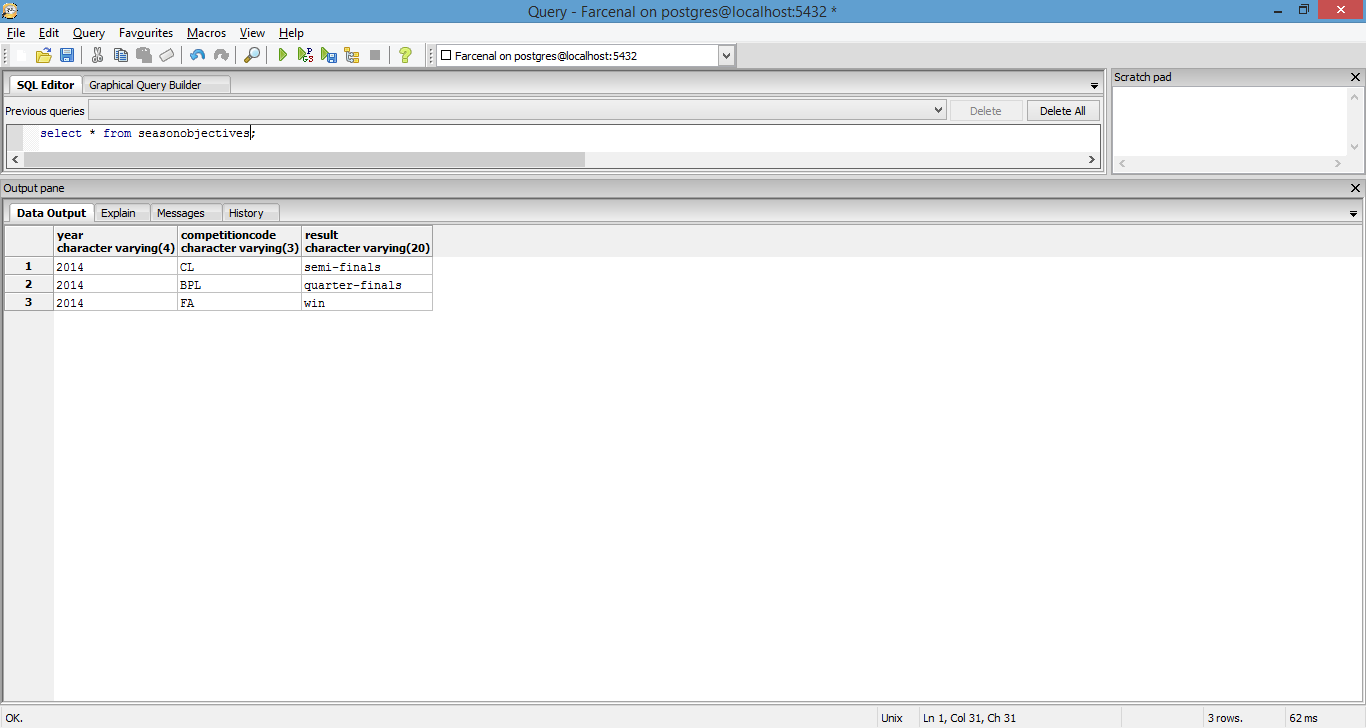
competitionCode varchar(3) NOT NULL,

result varchar(20) NOT NULL

constraint result CHECK (result = 'win' OR result = 'semi-finals' OR result = 'quarter-finals' OR result = 'round of 16')

);

**Sample data**



**injuryDetails table**

**Purpose:** Determine details and length of injury for various players in the squad.

**Functional dependencies**

PID -> injured, year, injury, recoveryDays

**Table create statement**

CREATE TABLE injuryDetails (

PID integer NOT NULL,

injured boolean,

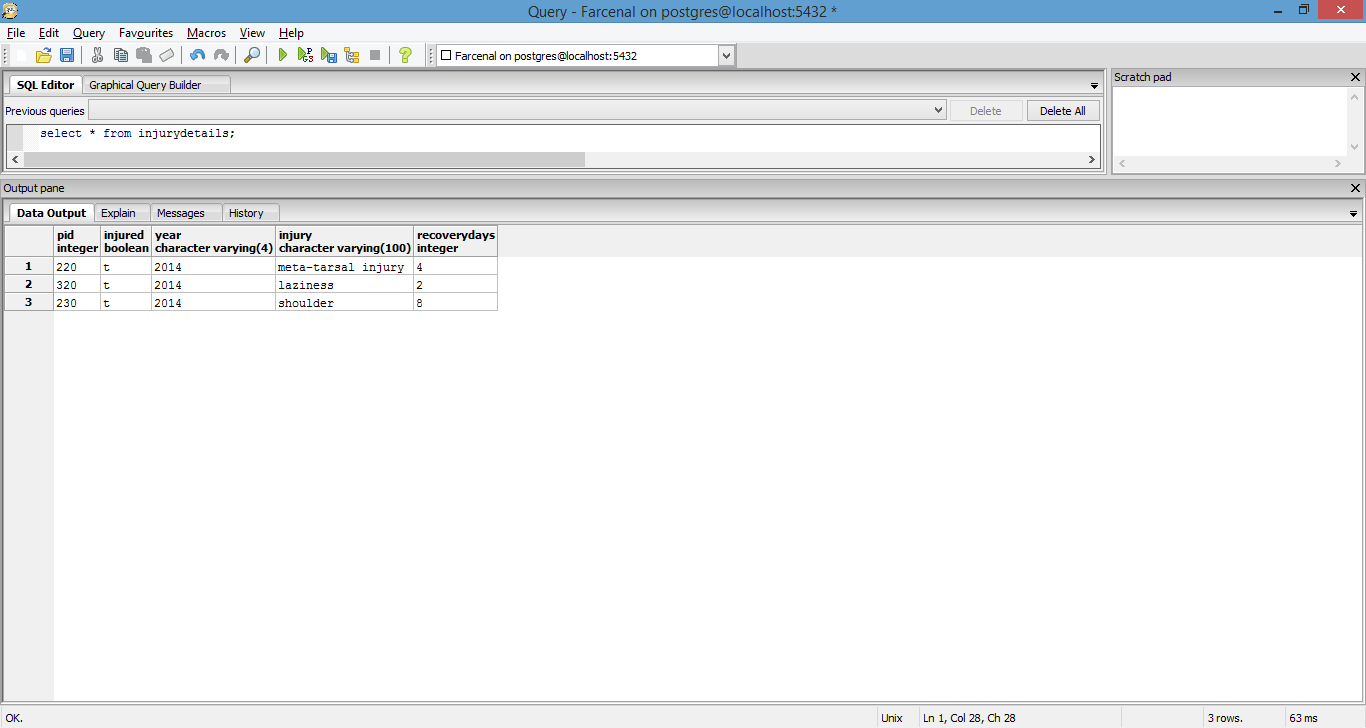
year varchar(4),

injury varchar(100),

recoveryDays integer

);

**Sample data**



**Specialty table**

**Purpose:** Details about the specialty of each of the doctors on staff.

**Functional dependencies**

specialtyCode -> specialtyDescr

**Table create statement**

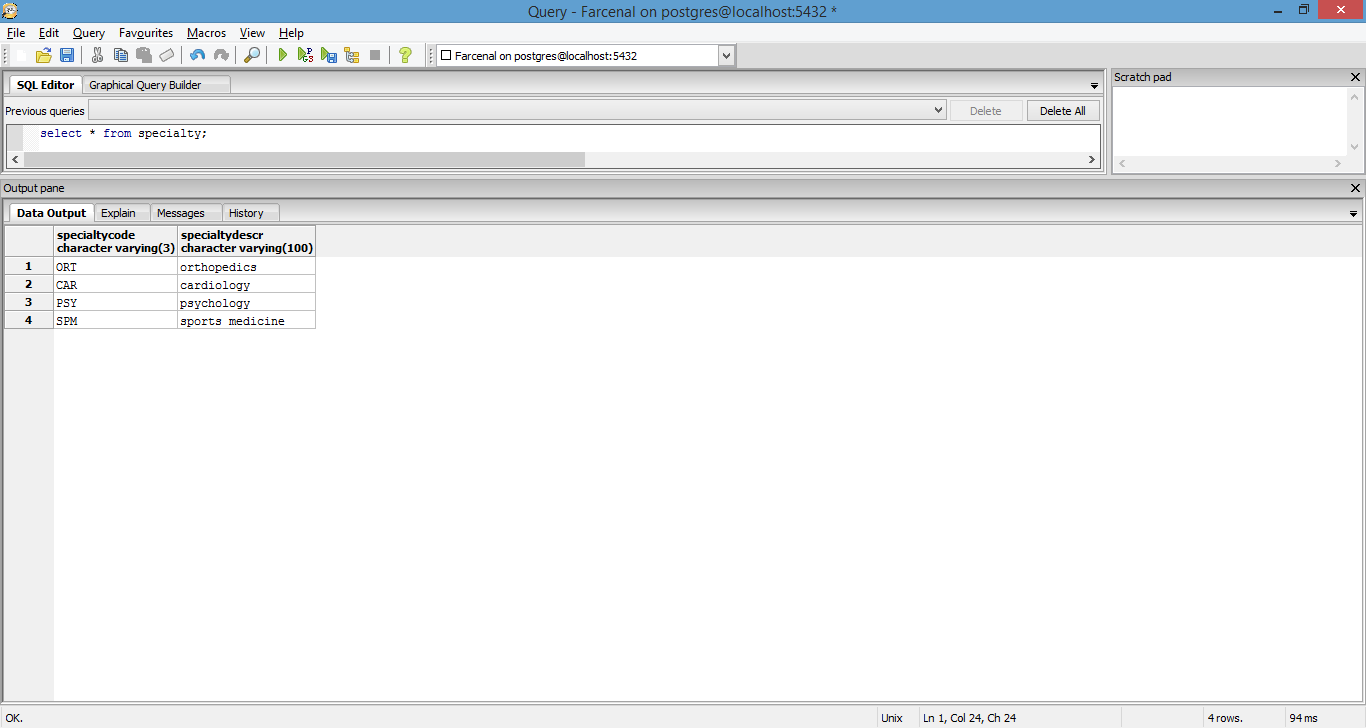
CREATE TABLE specialty (

specialtyCode varchar(3) PRIMARY KEY,

specialtyDescr varchar(100) NOT NULL

);

**Sample data**



**signonMedicals table**

**Purpose:** Gives information on whether certain scouted players passed their medical screening or not prior to signing for the club.

**Functional dependencies**

PID -> year, medicalPassed, screeningNum

**Table create statement**

CREATE TABLE signonMedicals (

PID integer NOT NULL,

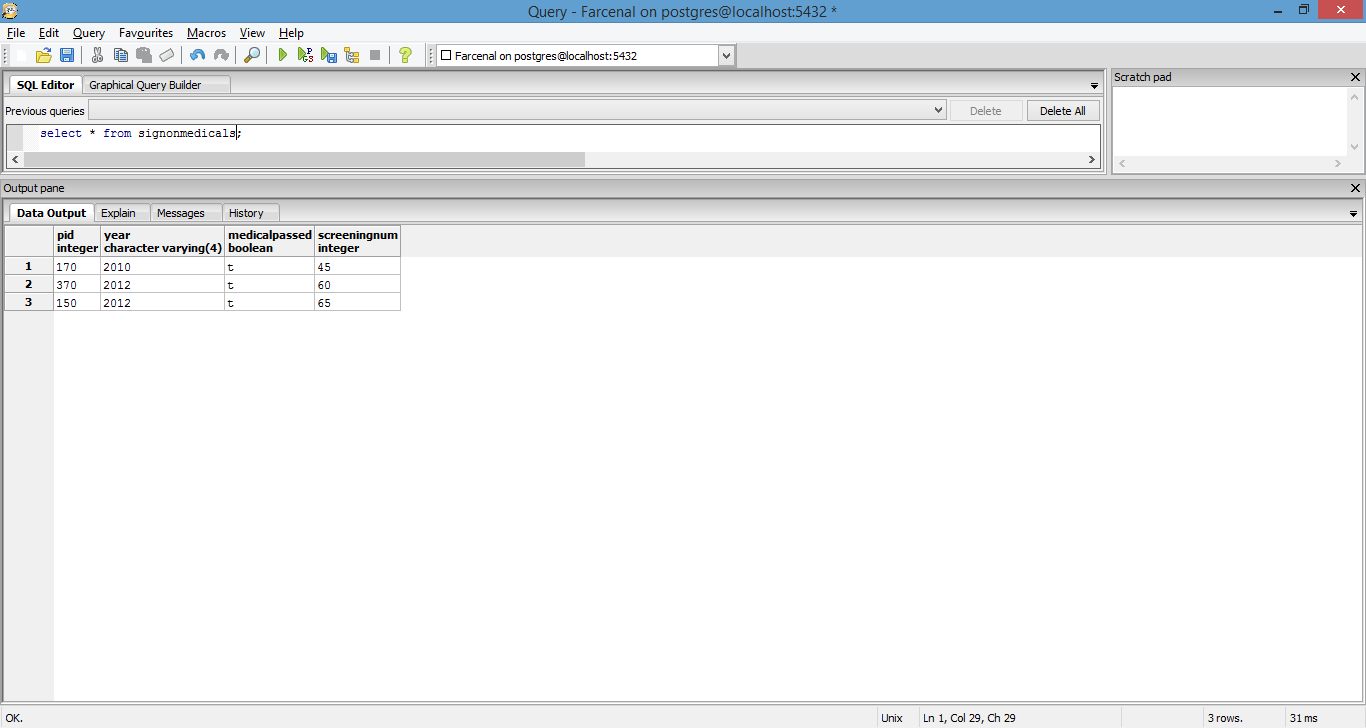
year varchar(4) NOT NULL,

medicalPassed boolean NOT NULL,

screeningNum integer NOT NULL

);

**Sample data**



**Screening table**

**Purpose:** Gives details about each screening listed in the table above

**Functional dependencies**

screeningNum -> details

**Table create statement**

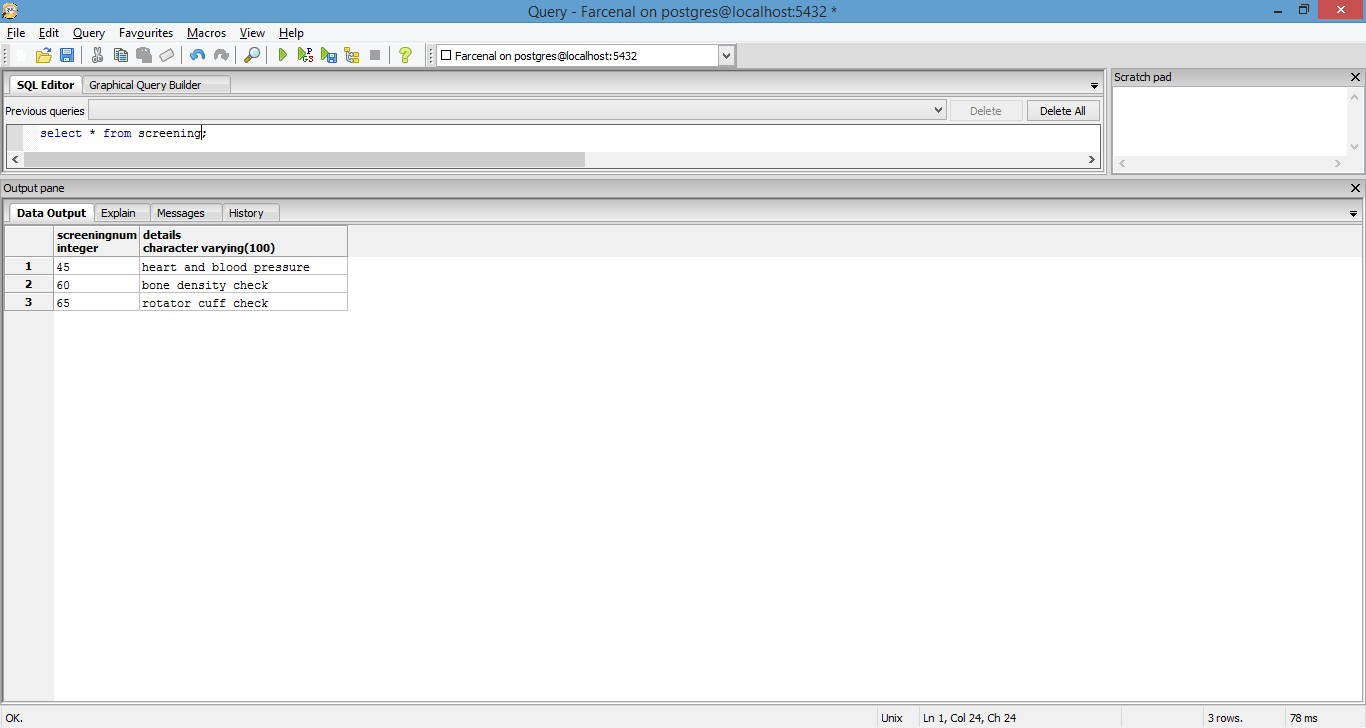
CREATE TABLE screening (

screeningNum integer PRIMARY KEY,

details varchar(100)

);

**Sample data**



**playerStats table**

**Purpose:** Snapshot of all statistics related to each player on the squad in a given season. Including disciplinary records, and goals scored per competition entered by the club in a given year.

**Functional dependencies**

PID -> yellowCards, redCards, goals, competitionCode, year, cupTied

**Table create statement**

CREATE TABLE playerStats (

PID integer NOT NULL,

yellowCards integer,

redCards integer,

goals integer,

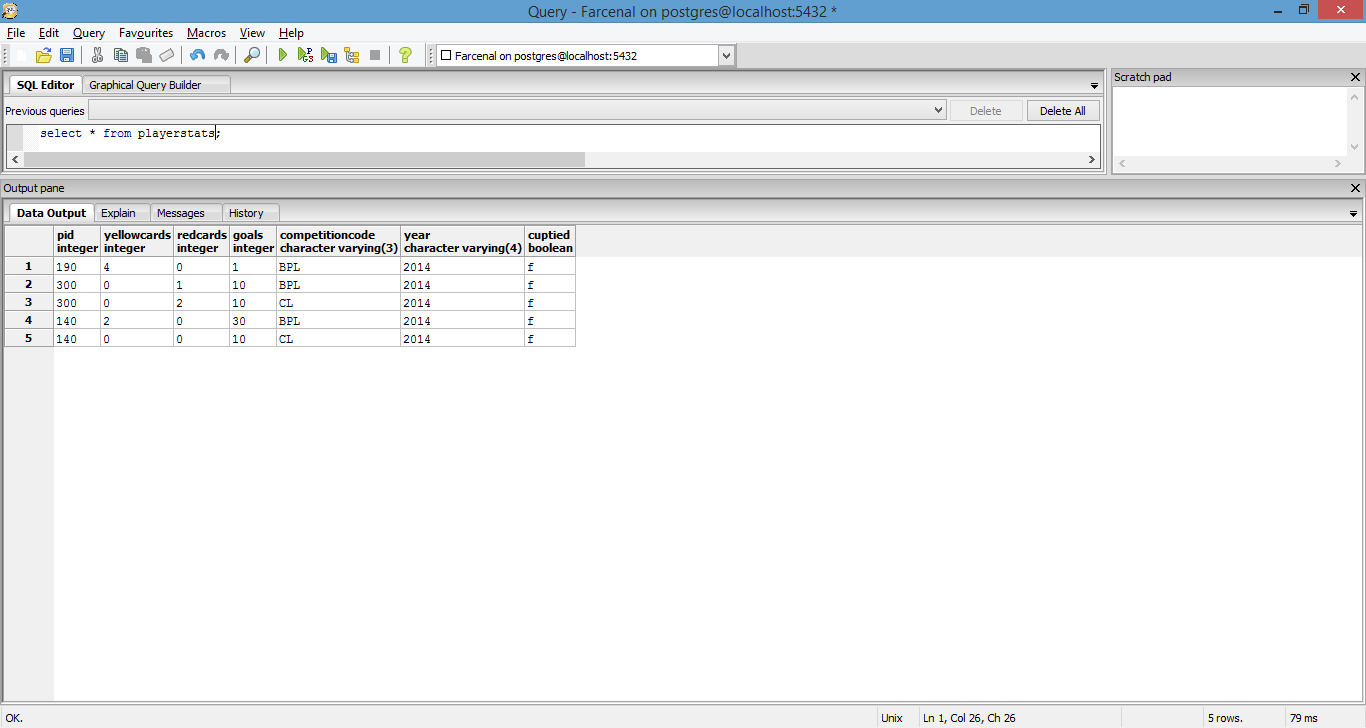
competitionCode varchar(3),

year varchar(4),

cupTied boolean

);

**Sample data**



**cupTied table**

**Purpose:** Helps determine whether a player is “cup tied” for a particular competition entered by the club. This will help the manager determine whether that player can be put on the roster. Usually this involves players that have been taken by the club “on loan” for a season from a competing club. They are generally not allowed to play their “parent club”.

**Functional dependencies**

PID -> competitionCode, year, clubCode

**Table create statement**

CREATE TABLE cupTied (

PID integer NOT NULL,

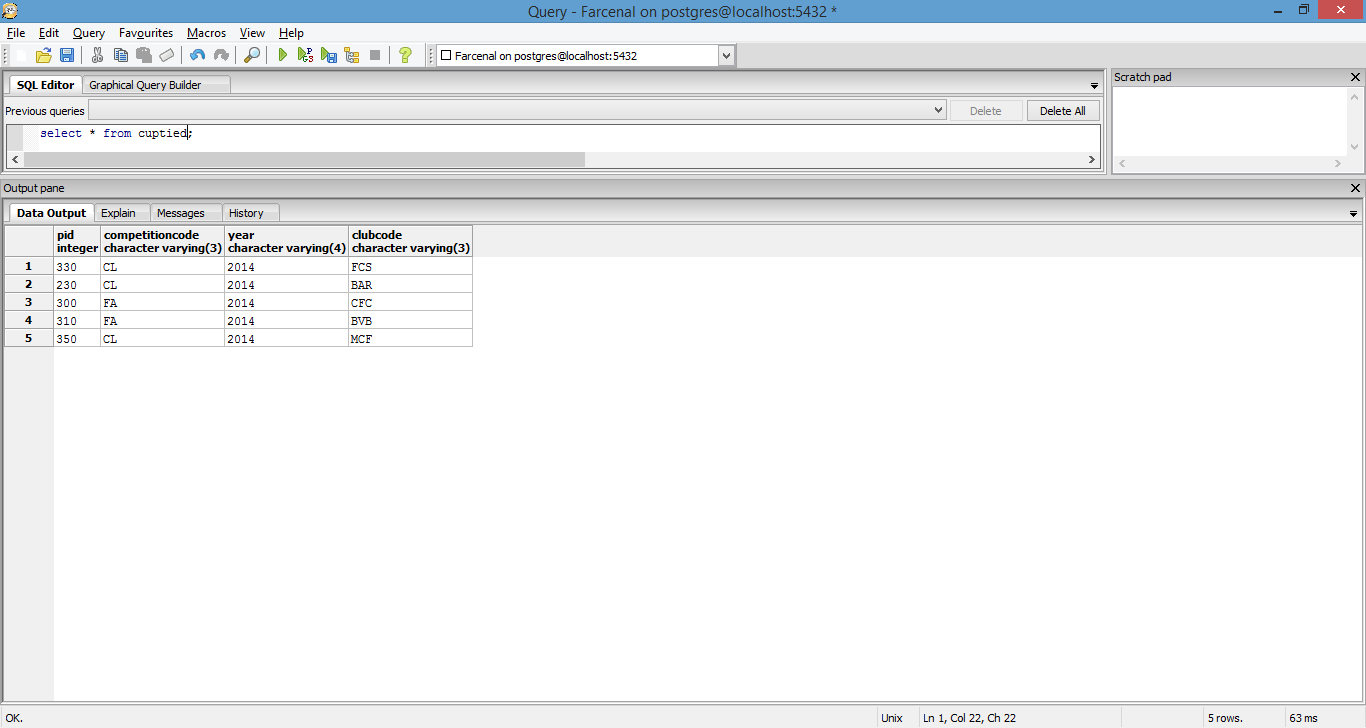
competitionCode varchar(3) NOT NULL,

year varchar(4) NOT NULL,

clubCode varchar(3) NOT NULL

);

**Sample data**



**Clubs table**

**Purpose:** Stores information on competing clubs. Generally used to populate information in the table above.

**Functional dependencies**

clubCode -> clubDescr

**Table create statement**

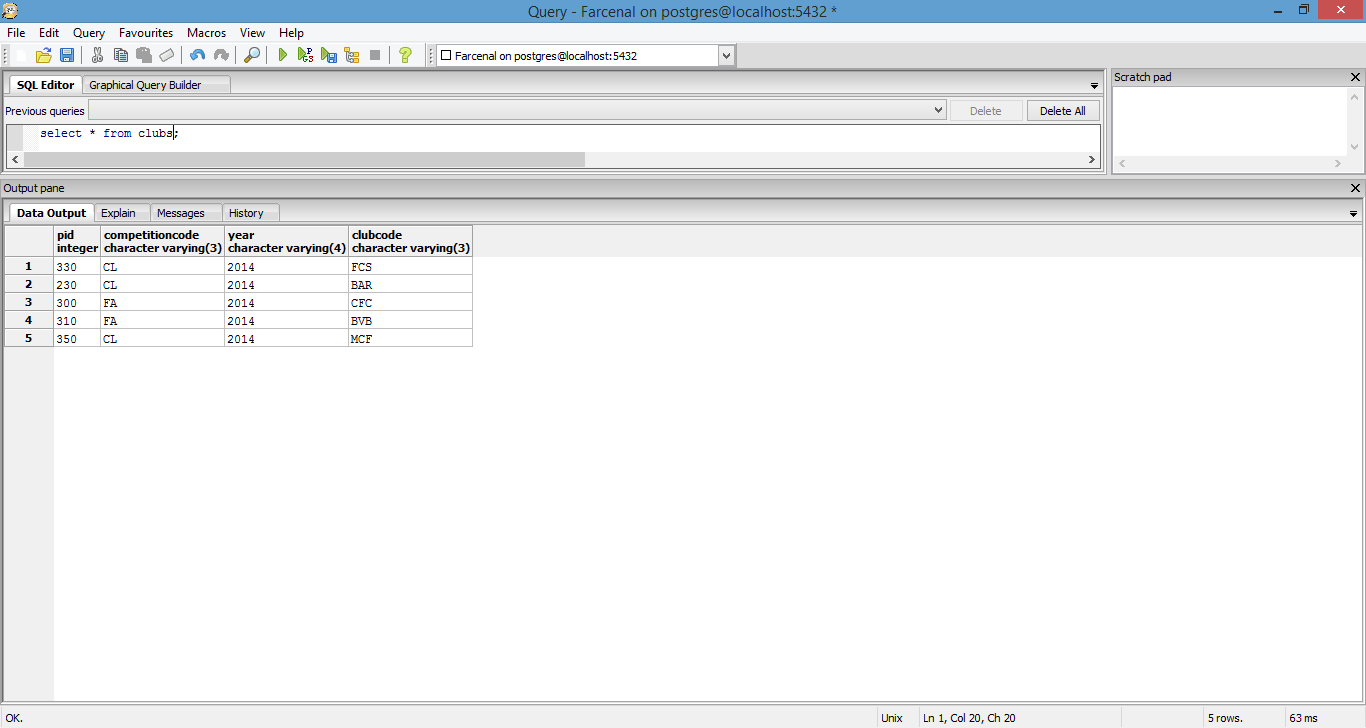
CREATE TABLE clubs (

clubCode varchar(3) PRIMARY KEY,

clubDescr varchar(100) NOT NULL

);

**Sample data**



**Views**

**PlayersByPosition**

**Purpose:** Returns listing of every player in the club (including first team, reserve team and youth team) along with a detailed description of the position they play in.

**Code**

CREATE VIEW PlayersByPosition AS

SELECT p.fname,

p.lname,

p.mi,

p.age,

pos.positiondescr

FROM players pl

INNER JOIN people p on

pl.pid = p.pid

INNER JOIN positions pos on

pl.positioncode = pos.positioncode

**PlayersByCountry**

**Purpose:** As mentioned under the table create scripts above, the club can have employees from several nations. This view was created with the purpose of determining which nationality each employee is from in case HR needs a report on this or for work permit purposes.

**Code**

CREATE VIEW PlayersByCountry AS

SELECT p.fname,

p.lname,

p.mi,

p.age,

p.countrycode,

c.countryName

FROM players pl

INNER JOIN people p on

pl.pid = p.pid

INNER JOIN countries c on

p.countryCode = c.countryCode

**PlayersByInjury**

**Purpose:** Can be used while generating injury reports for the manager by the medical team.

**Code**

CREATE VIEW PlayersByInjury AS

SELECT p.fname,

p.lname,

p.mi,

p.age,

pl.positioncode,

i.injury,

i.recoveryDays

FROM players pl

INNER JOIN people p on

pl.pid = p.pid

INNER JOIN injuryDetails i on

pl.pid = i.pid

WHERE i.injured = true

**SponsorDetails**

**Purpose:** Easy way to represent summary information related to sponsors and related sponsorships

**Code**

CREATE VIEW SponsorDetails AS

SELECT s.sponsordescr,

s.sponsornetworth,

sp.year,

sp.amount,

sp.type

FROM sponsors s

INNER JOIN sponsorships sp on

s.sponsorcode = sp.sponsorcode

**WageDetails**

**Purpose:** Easy way to represent summary information related to employee wages if a payroll report needs to be generated.

**Code**

CREATE VIEW WageDetails AS

SELECT p.fname,

p.mi,

p.lname,

p.etid,

w.annualsalary,

c.competitiondescr,

w.competitionbonus,

w.bonusamount,

w.year

FROM wages w

INNER JOIN people p on

w.pid = p.pid

INNER JOIN competitions c on

w.competitionCode = c.competitionCode

**PlayersInInternationalTournaments**

**Purpose:** Club players can come from several nations. Due to this, their national teams may call them up for duty either during or in between seasons. Managers will like to monitor this information in case a club player picks up an injury while on national duty.

**Code**

CREATE VIEW PlayersInInternationalTournaments AS

SELECT p.fname,

p.mi,

p.lname,

ic.tournamentdescr,

ic.year

FROM people p

INNER JOIN players pl on

p.pid = pl.pid

INNER JOIN internationalcompetitions ic on

p.countryCode = ic.countryCode

**EmployeeDetails**

**Purpose:** Information on the type of employee each person is.

**Code**

CREATE VIEW EmployeeDetails AS

SELECT p.fname,

p.mi,

p.lname,

et.etdescr

FROM people p

INNER JOIN employeeType et on

p.etid = et.etid

**Reports**

**GetInjuryListing**

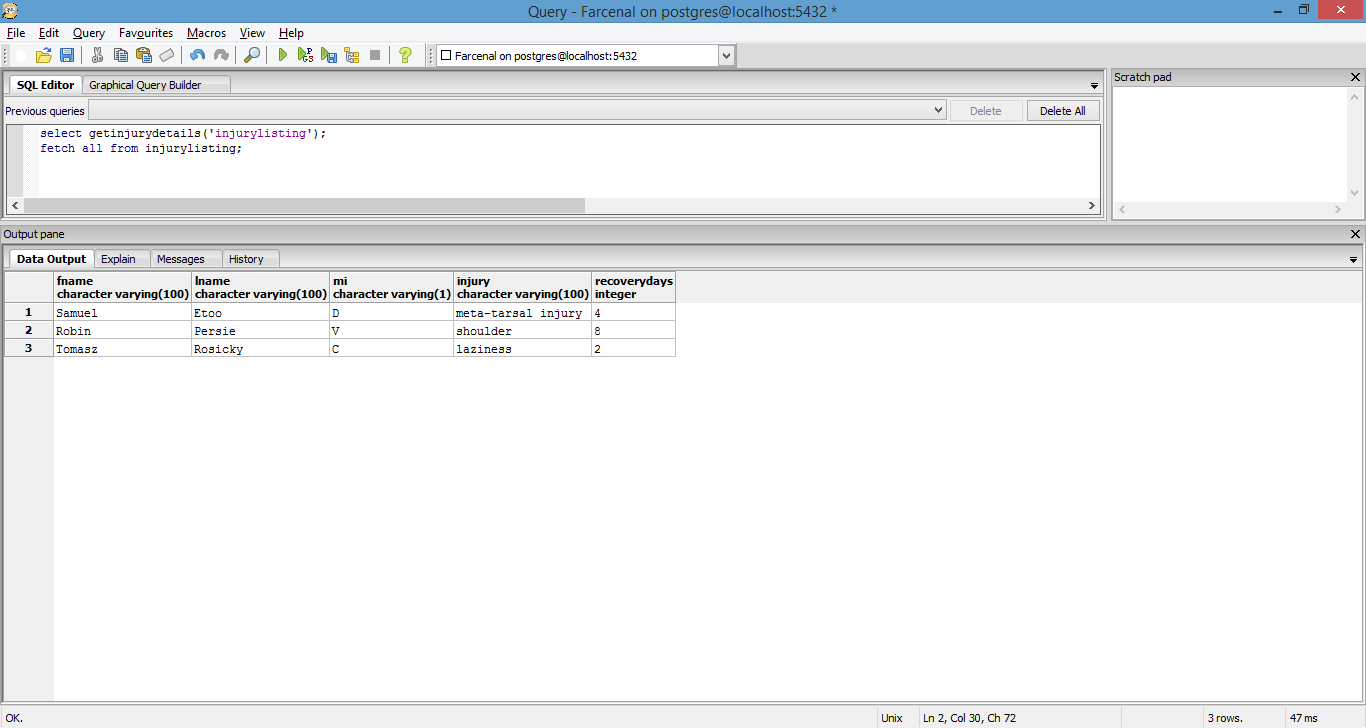
**Purpose:** To aid the manager in making team selections.

**Code**

select getinjurydetails('injurylisting');

fetch all from injurylisting;

**Sample**

****

**GetWorkPermitListing**

**Purpose:** To aid the manager or human resources.

**Code**

--need work permit

select getworkpermitdetails(true,'workpermit');

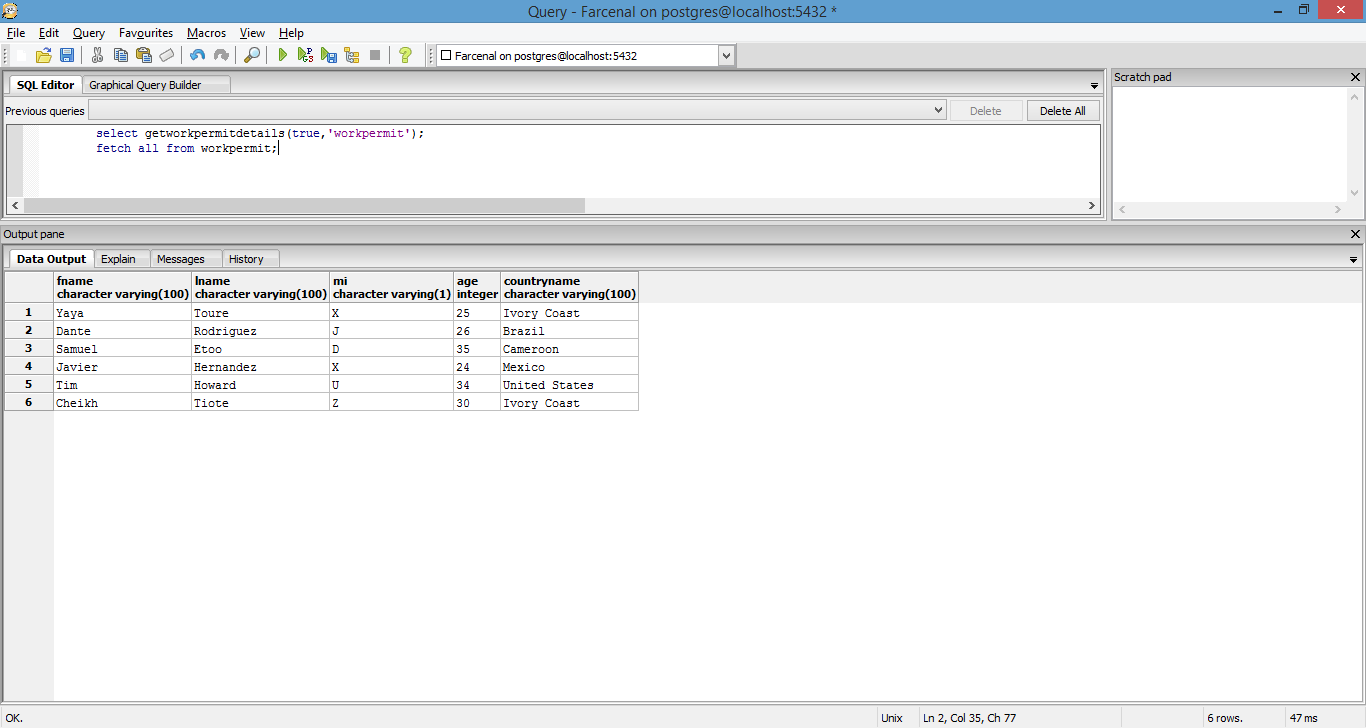
fetch all from workpermit;

--don't need work permit

select getworkpermitdetails(false,'workpermit');

fetch all from workpermit;

**Sample**



**GetSponsorDetails**

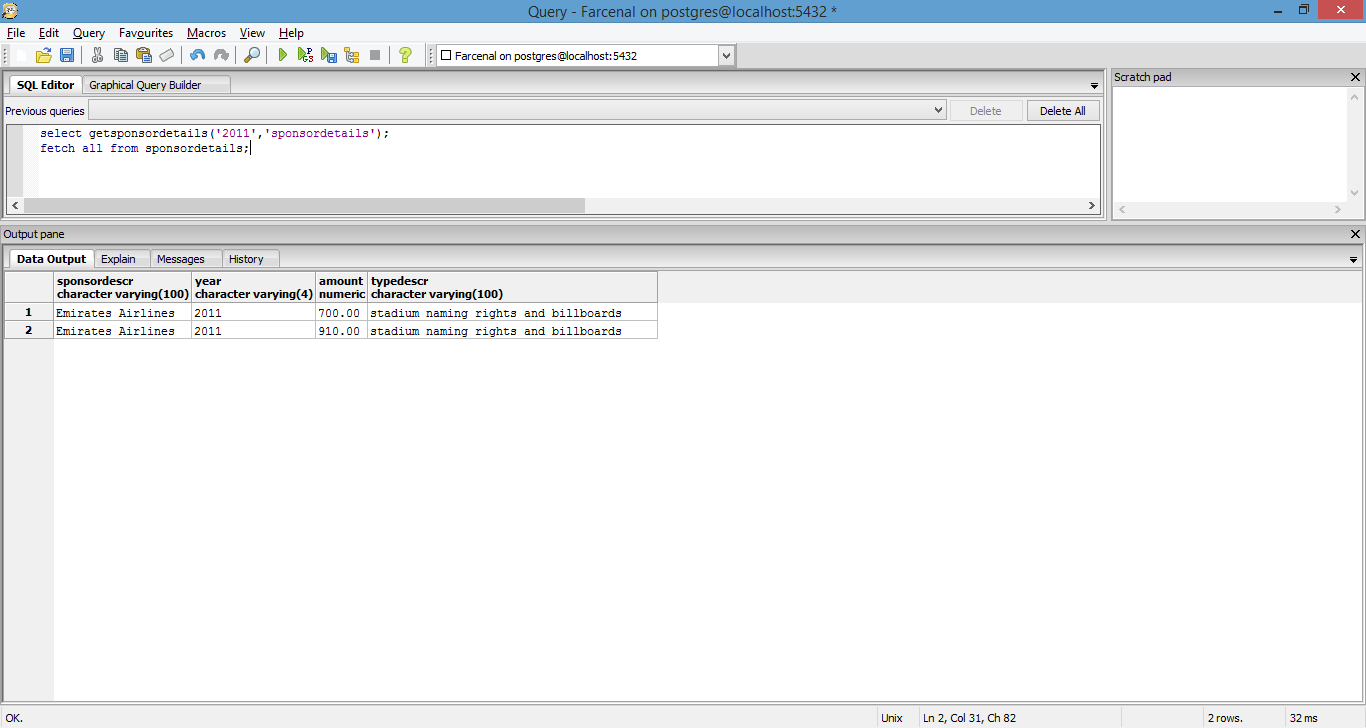
**Purpose:** To provide management, finance department and owners a snapshot of club sponsorship information.

**Code**

select getsponsordetails('2011','sponsordetails');

fetch all from sponsordetails;

**Sample**



**TournamentParticipation**

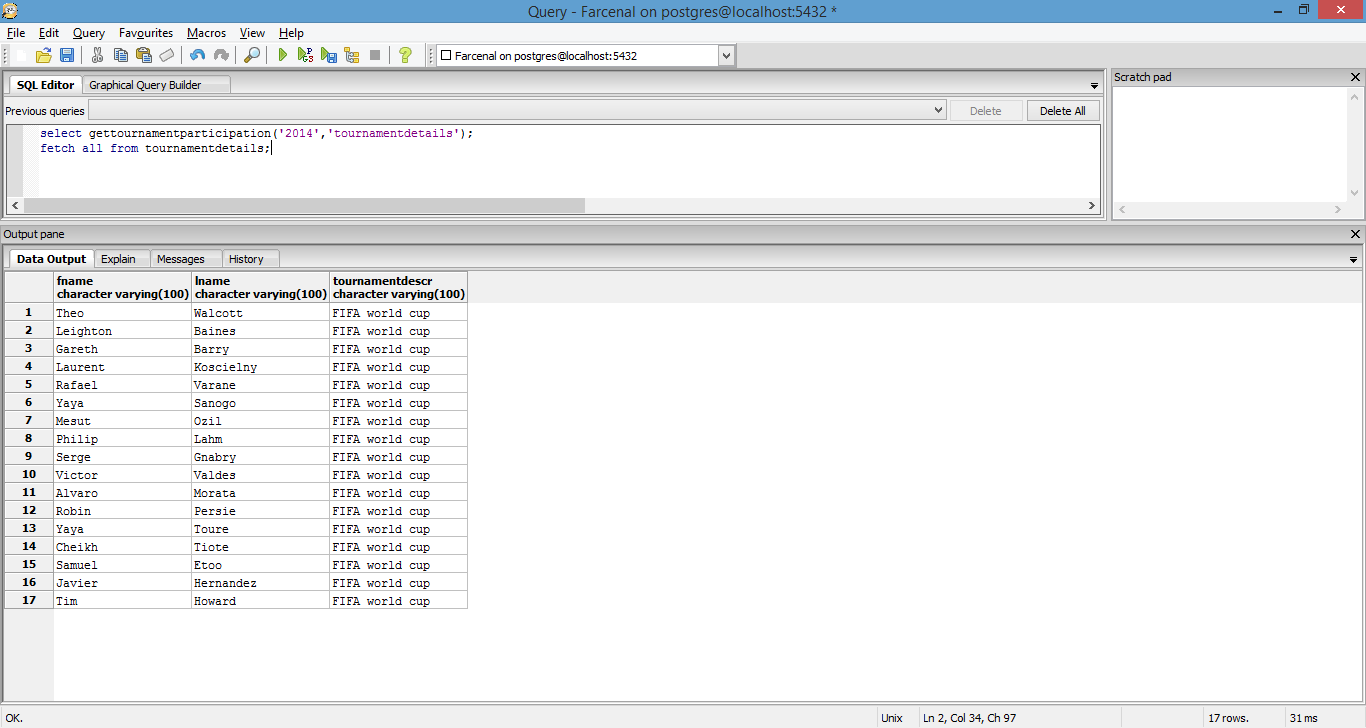
**Purpose:** To aid the manager in making team selections based on whether or not some of his players will be away at the tournament(s) in question.

**Code**

select gettournamentparticipation('2014','tournamentdetails');

fetch all from tournamentdetails;

**Sample**



**EmployeeDetails**

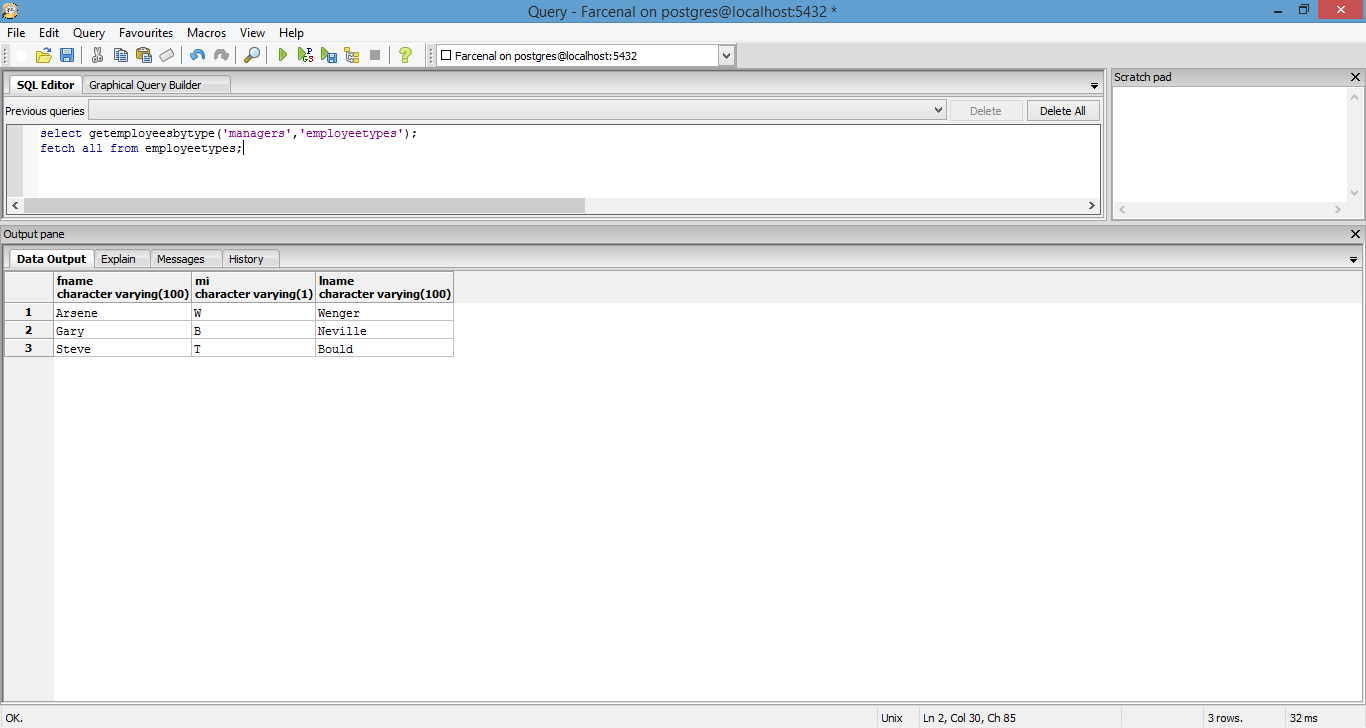
**Purpose:** To aid human resources to find details about employees of a certain category.

**Code**

select getemployeesbytype('managers','employeetypes');

fetch all from employeetypes;

**Sample**

****

**Stored Procedures**

**GetInjuryDetails**

**Purpose:** Complete listing of injured players including the injury description and recovery time needed.

create or replace function GetInjuryDetails(REFCURSOR) returns refcursor as

$$

declare

injurydetails REFCURSOR := $1;

begin

open injurydetails for

select fname,

lname,

mi,

injury,

recoverydays

from PlayersByInjury;

return injurydetails;

end;

$$

language plpgsql;

**GetWorkPermitDetails**

**Purpose:** To be used by HR to determine/obtain UK work permits for employees at the club that may need it.

create or replace function GetWorkPermitDetails(boolean, REFCURSOR) returns refcursor as

$$

declare

permitneeded boolean := $1;

workpermitdetails REFCURSOR := $2;

begin

open workpermitdetails for

select pbc.fname,

pbc.lname,

pbc.mi,

pbc.age,

pbc.countryname

from PlayersByCountry pbc

inner join workpermit w on

pbc.countrycode = w.countrycode

where w.ukworkpermit = permitneeded;

return workpermitdetails;

end;

$$

language plpgsql;

**GetTournamentParticipation**

**Purpose:** To determine what tournament(s) club players are involved in in a given year.

create or replace function GetTournamentParticipation(varchar(4), REFCURSOR) returns refcursor as

$$

declare

tournamentyear varchar(4) := $1;

tournamentdetails REFCURSOR := $2;

begin

open tournamentdetails for

select pit.fname,

pit.lname,

pit.tournamentdescr

from PlayersInInternationalTournaments pit

where pit.year = tournamentyear;

return tournamentdetails;

end;

$$

language plpgsql;

**GetSponsorDetails**

**Purpose:** To generate reports about sponsorship deals for the club.

create or replace function GetSponsorDetails(varchar(4), REFCURSOR) returns refcursor as

$$

declare

sponsoryear varchar(4) := $1;

sponsordetails REFCURSOR := $2;

begin

open sponsordetails for

select sd.sponsordescr,

sd.year,

sd.amount,

st.typedescr

from SponsorDetails sd

inner join sponsorType st on

sd.type = st.type

where sd.year = sponsoryear;

return sponsordetails;

end;

$$

language plpgsql;

**GetInjuryDetailsByPos**

**Purpose:** To determine squad selection

create or replace function GetInjuryDetailsByPos(varchar(3),REFCURSOR) returns refcursor as

$$

declare

injurypos varchar(4) := $1;

injurydetails REFCURSOR := $2;

begin

open injurydetails for

select fname,

lname,

mi,

injury,

recoverydays

from PlayersByInjury

where positioncode = injurypos;

return injurydetails;

end;

$$

language plpgsql;

**GetEmployeesByType**

**Purpose:** To generate employee reports

create or replace function GetEmployeesByType(varchar(100),REFCURSOR) returns refcursor as

$$

declare

etype varchar(100) := $1;

employeetype REFCURSOR := $2;

begin

open employeetype for

select fname,

mi,

lname

from EmployeeDetails

where etdescr = etype;

return employeetype;

end;

$$

language plpgsql;

**Triggers**

**Check\_PrimaryAddress()**

**Purpose:** While inserting address information, a PID can only list one address as the “primary”. This checks for that.

CREATE FUNCTION check\_primaryAddress() RETURNS TRIGGER AS $check\_primaryAddress$

BEGIN

IF

EXISTS (SELECT Addr1

FROM address

WHERE PID = NEW.PID

AND NEW.IsPrimary = true)

THEN

RAISE EXCEPTION 'Cannot have more than one address listed as primary for a PID';

END IF;

RETURN NEW;

END

$check\_primaryAddress$ LANGUAGE plpgsql;

**Check\_isPlayer()**

**Purpose:** To make sure the right type ­of employee is being entered into the right table. **Example:** A player cannot be a doctor and therefore a PID of type player should not be inserted into the doctor table.

CREATE FUNCTION check\_isPlayer() RETURNS TRIGGER AS $check\_isPlayer$

BEGIN

IF

EXISTS (SELECT PID

FROM players

WHERE PID = NEW.PID)

THEN

RAISE EXCEPTION 'A player cannot assume a role other than being in the main team, reserve team, or youth team';

END IF;

RETURN NEW;

END

$check\_isPlayer$ LANGUAGE plpgsql;

**Check\_isDoctor()**

**Purpose:** Very similar to above.

CREATE FUNCTION check\_isDoctor() RETURNS TRIGGER AS $check\_isDoctor$

BEGIN

IF

EXISTS (SELECT PID

FROM doctors

WHERE PID = NEW.PID)

THEN

RAISE EXCEPTION 'A doctor cannot be on the playing squad';

END IF;

RETURN NEW;

END

$check\_isDoctor$ LANGUAGE plpgsql;

**Security**

Based on the current design of this system, there would be the following types of user accounts (not counting the super user/DBA)

**Managers:**

Would only be able to access tables, views and reports associated with the following aspects of the clubs

* Squad players
* Youth team players
* Reserve team players
* Scouting information
* Scouting history
* Depending on the type of responsibilities given to the manager by the owners of the club, a manager may be able to access financial and sponsorship tables, views and reports as well

**Human resources:**

Would only be able to access tables, views and reports associated with the following aspects of the system

* Employee types
* Whether or not employees require work permits
* Wage information

**Payroll:**

Would be able to see higher level wage information including whether or not employees get performance based bonuses and details pertaining to the same

**Club owners:**

Will be able to access all tables, views and reports within the system in order to properly manage the club. However, access will be limited to querying and reports

**Medical staff:**

Will be able to access health and injury related tables, views and stored procedures as well as tables, views and stored procedures related to player health history.

**Implementation notes/known problems/future enhancements**

This implementation was meant to be a relatively simple representation of a select few operations of a professional football club. However, given the nature of football clubs and considering the vastly complicated nature of their day to day operations, the scope of the project grew very quickly. More tables were needed (compared to the initial estimate) in order to properly represent how this form of system would work even with a small amount of sample data. A recommended enhancement would be to create a system that allows the representation of several more employee types considering that a professional club would have various types of employees other than just managers, players, coaches, and doctors. For example, nutritionists play a very important role in athletic facilities. This system could be expanded to include detailed information related to nutrition for the players and nutritionists. Another suggested enhancement would be to create more triggers, specifically those related to role checking. Right now, the existing triggers are very basic in nature and don’t consider the complex relationships that could exist between employee types. For instance, a PID could potentially be a player and a coach, or a coach and a doctor (if we consider staff members like physical therapists).