Football Team Management

DBMS project

Sabrina Lupsan

Faculty of Cybernetics, Statistics and Economic Informatics

Coordinating professor: Diaconita Vlad

Series G

Group 1064

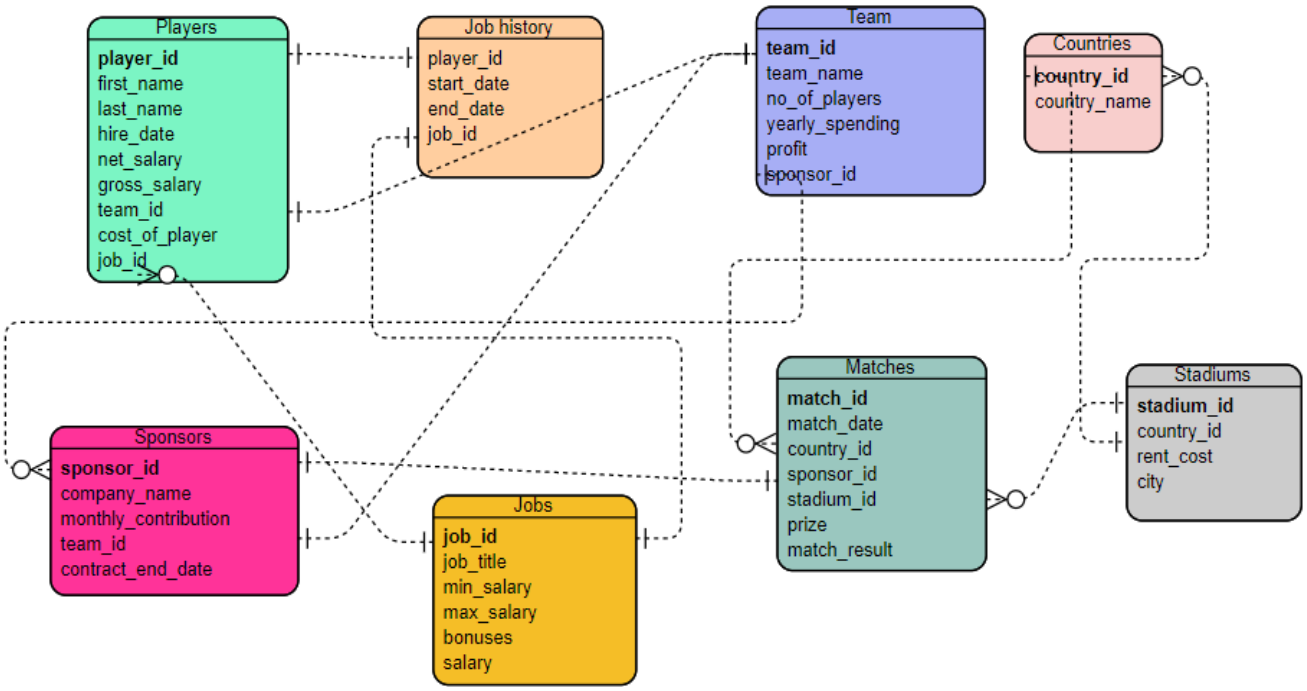
Description of the theme

I chose to make a database of a Football Team and learned how to manage it. The players have four different kind of jobs, based on real-life evaluation and strategies – goalkeeper, midfielder, defender and attacker. The minimum and maximum salary of each job depend on the job and the net salaries were calculated using the gross salary and the taxes applied. Each team has players, a sponsor, matches played. The matches were played in different stadiums, and those stadiums are in different countries. This is how it all ties together, bringing the management of a Football Team. I believe that such a database is useful for storing data for official competitions like Champions League or The World Championship, which need official data, and for media news, who need statistics, such as the ones displayed in the exercises given. There are large volumes of data in this field because there are hundreds of matches per week all around the world.

All of the teams and names were fictious and the salaries chosen for this database are not official or true.

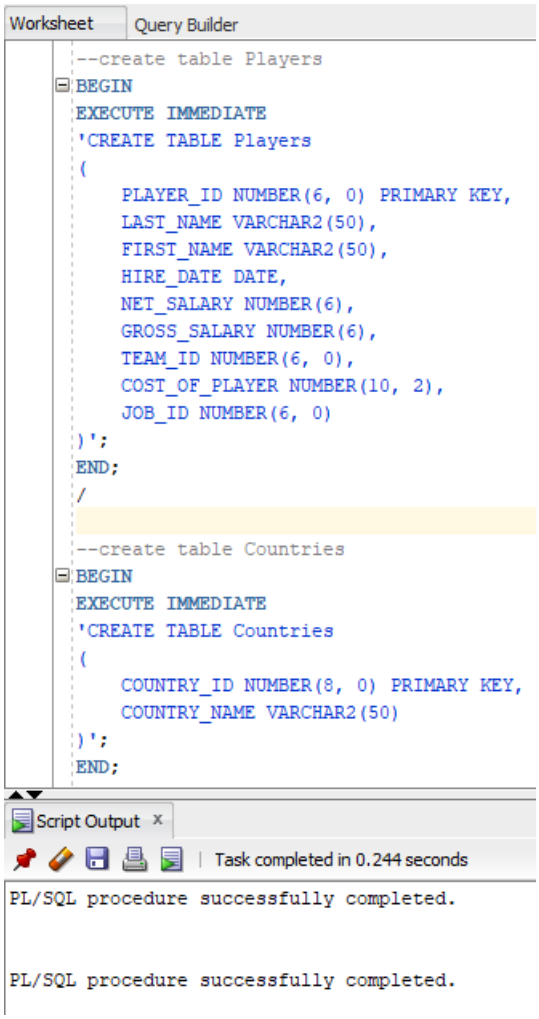
1. The conceptual schema of the database

A database always begins with a conceptual schema, which helps to design and organize the coding part. This also helps identify **the unique keys** (in bold), foreign keys, connections to other tables in the database and, eventually, errors.



2. DDL and DML statements

```
--create table Players
BEGIN
EXECUTE IMMEDIATE
'CREATE TABLE Players
(
   PLAYER_ID NUMBER(6, 0) PRIMARY KEY,
   LAST_NAME VARCHAR2(50),
   FIRST_NAME VARCHAR2(50),
   HIRE_DATE DATE,
   NET_SALARY NUMBER(6),
   GROSS_SALARY NUMBER(6),
   TEAM_ID NUMBER(6, 0),
   COST_OF_PLAYER NUMBER(10, 2),
   JOB_ID NUMBER(6, 0)
)';
END;
/
```




```
--create table Countries

BEGIN

EXECUTE IMMEDIATE

'CREATE TABLE Countries

(

   COUNTRY_ID NUMBER(8, 0) PRIMARY KEY,

   COUNTRY_NAME VARCHAR2(50)

)';

END;
```

```
/

--create table Stadiums
BEGIN
EXECUTE IMMEDIATE
'CREATE TABLE Stadiums
(
   STADIUM_ID NUMBER(8, 0) PRIMARY KEY,
   COUNTRY_ID NUMBER(8, 0),
   RENT_COST NUMBER(5, 2),
   CITY VARCHAR2(20)
)';
END;
/


--create table Team
BEGIN
EXECUTE IMMEDIATE
'CREATE TABLE Team
(
   TEAM_ID NUMBER(6, 0) PRIMARY KEY,
   TEAM_NAME VARCHAR2(100),
   NUMBER_OF_PLAYERS NUMBER(3),
   YEARLY_SPENDINGS NUMBER(8, 2),
```

```
   PROFIT NUMBER(5, 2),
   SPONSOR_ID NUMBER(5, 2)
)';
END;
/

--create table Jobs
BEGIN
EXECUTE IMMEDIATE
'CREATE TABLE Jobs
(
   JOB_ID NUMBER(8, 0) PRIMARY KEY,
   JOB_TITLE VARCHAR2(50),
   MIN_SALARY NUMBER(8, 2),
   MAX_SALARY NUMBER(8, 2),
   BONUSES NUMBER(6, 2),
   SALARY NUMBER(8, 2)
)';
END;
/

--create table Job_History
BEGIN
EXECUTE IMMEDIATE
'CREATE TABLE Job_History
```

```
(
   JOB_ID NUMBER(8, 0),
   START_DATE DATE,
   END_dATE DATE,
   PLAYER_ID NUMBER(8, 0)
)';
END;
/

--create table Sponsors
BEGIN
EXECUTE IMMEDIATE
'CREATE TABLE Sponsors
(
   SPONSOR_ID NUMBER(8, 0) PRIMARY KEY,
   COMPANY_NAME VARCHAR2(200),
   MONTHLY_CONTRIBUTION NUMBER(6,2),
   TEAM_ID NUMBER(6),
   CONTRACT_END_DATE DATE
)';
END;
/

--create table Match
BEGIN
```

```
EXECUTE IMMEDIATE
'CREATE TABLE Match
(
   MATCH_ID NUMBER(8, 0) PRIMARY KEY,
   MATCH_DATE DATE,
   COUNTRY_ID NUMBER(8,0),
   SPONSOR_ID NUMBER(8,0),
   PRIZE NUMBER(5, 0),
   MATCH_RESULT VARCHAR(50)
)';
END;
/

--rename the table "Match" to "Matches"
BEGIN
EXECUTE IMMEDIATE
'ALTER TABLE Match RENAME TO Matches';
END;
/

--drop the column "FIRST_NAME" from the Players table
BEGIN
EXECUTE IMMEDIATE
'ALTER TABLE Players DROP COLUMN FIRST_NAME';
END;
```

/

--update the net salary of each player by deducting social (25%) and health (10%) insurance

--and the income tax (10%) from the gross salary

```
BEGIN
UPDATE PLAYERS
SET NET_SALARY = 0.9*(GROSS_SALARY - 0.35*GROSS_SALARY);
END;
/
```

--divide the players into 2 teams: players with the player_id in the interval [1, 2, 3, 4] are in the

--team with the id=5555 and players with IDs 5 and 6 are in the team with the team_id = 6666

```
BEGIN
UPDATE PLAYERS
SET TEAM_ID = 5555 WHERE PLAYER_ID IN (1, 2, 3, 4);
UPDATE PLAYERS
SET TEAM_ID = 6666 WHERE PLAYER_ID IN (5, 6);
END;
/
```

--Display in how much time the contract of every sponsor will expire, in intervals of: <1 year, 1-2 years, > 2 years.

```
SET SERVEROUTPUT ON
DECLARE
CURSOR end_date IS SELECT company_name,
(CASE
```

```
   WHEN EXTRACT(YEAR FROM contract_end_date) - EXTRACT(YEAR FROM SYSDATE) < 1 THEN 'Less
than 1 year'

   WHEN EXTRACT(YEAR FROM contract_end_date) - EXTRACT(YEAR FROM SYSDATE) BETWEEN 1 AND 2
THEN 'Between 1 and 2 years'

   ELSE 'More than 2 years'

END) YEARS

FROM Sponsors

ORDER BY (EXTRACT(YEAR FROM contract_end_date));

company end_date%rowtype;

BEGIN

OPEN end_date;

LOOP

   FETCH END_DATE INTO company;

   EXIT WHEN END_DATE%NOTFOUND;

   DBMS_OUTPUT.PUT_LINE(COMPANY.COMPANY_NAME||' -> '||COMPANY.YEARS);

END LOOP;

CLOSE end_date;

END;

/
```

0.51899999 seconds

Worksheet | Query Builder

```
--Display in how much time the contract of every sponsor will expire, in intervals of: <1 year, 1-2 years, > 2 years.
SET SERVEROUTPUT ON
DECLARE
CURSOR end_date IS SELECT company_name,
(CASE
    WHEN EXTRACT(YEAR FROM contract_end_date) - EXTRACT(YEAR FROM SYSDATE) < 1 THEN 'Less than 1 year'
    WHEN EXTRACT(YEAR FROM contract_end_date) - EXTRACT(YEAR FROM SYSDATE) BETWEEN 1 AND 2 THEN 'Between 1 and 2 years'
    ELSE 'More than 2 years'
END) YEARS
FROM Sponsors
ORDER BY (EXTRACT(YEAR FROM contract_end_date));
company end_date%rowtype;
BEGIN
OPEN end_date;
LOOP
    FETCH END_DATE INTO company;
    EXIT WHEN END_DATE%NOTFOUND;
--FOR company in end_date LOOP
    DBMS_OUTPUT.PUT_LINE(COMPANY.COMPANY_NAME||' -> '||COMPANY.YEARS);
END LOOP;
END;
/
```

Script Output | Task completed in 0.519 seconds

```
PL/SQL procedure successfully completed.

Sponsor1 -> Between 1 and 2 years
Sponsor2 -> Between 1 and 2 years
Sponsor3 -> More than 2 years
```

--compute the value of the net salaries that a team would have to pay if it would only have players with even IDs

```
SET SERVEROUTPUT ON

DECLARE

type player is record (

   player_id number(8, 2),

   net_salary number(10, 2)

   );

v_player player;

v_total number(10, 0):=0;

v_players number(4, 0):=0;

BEGIN
```

```
SELECT COUNT(PLAYER_ID) INTO v_players FROM PLAYERS;

for i in 1..v_players loop

   if i mod 2 = 0 then

      select player_id, net_salary into v_player from players where player_id = i;

      v_total := v_total + v_player.net_salary;

   end if;

end loop;

dbms_output.put_line('The net salary of players with even player_id is: '||v_total);

end;

/


--display the name and player ID of the player in the 'Rangers' team. If there is more than one, print that

SET SERVEROUTPUT ON

DECLARE

v_name VARCHAR2(50);

v_id NUMBER(8);

BEGIN

SELECT last_name, player_id INTO v_name, v_id FROM PLAYERS WHERE TEAM_ID = (SELECT TEAM_ID
FROM TEAM WHERE team_name = 'Rangers');

EXCEPTION

   WHEN TOO_MANY_ROWS THEN

      DBMS_OUTPUT.PUT_LINE('There is more than one player');

END;

/


SET SERVEROUTPUT ON
```

```
DECLARE

CURSOR CITIES IS SELECT STADIUM_ID, CITY FROM STADIUMS;

acity CITIES%ROWTYPE;

BEGIN

FOR acity in CITIES LOOP

   OPEN CITIES;

   DBMS_OUTPUT.PUT_LINE('Stadium '||acity.stadium_id||' is in: '||acity.CITY);

END LOOP;

EXCEPTION WHEN CURSOR_ALREADY_OPEN THEN

   DBMS_OUTPUT.PUT_LINE('There was an error handling the cursor.');

END;

/
```

--Display the matches with the prize bigger than a given value dollars but exclude the ones in which the guest team did not win or they tied.

```
SET SERVEROUTPUT ON

DECLARE

CURSOR PRIZES(p_prize number) IS SELECT MATCH_ID, PRIZE, MATCH_DATE, MATCH_RESULT FROM
MATCHES WHERE prize > p_prize;

v_prize prizes%rowtype;

BEGIN

FOR v_prize in prizes(10000) LOOP

   if v_prize.match_result <> 'Host team lost' THEN

      DBMS_OUTPUT.PUT_LINE(v_prize.match_id||' '||v_prize.prize||' '||v_prize.match_date||'
'||v_prize.match_result);

   END IF;

END LOOP;
```

```
END;
/

--display all jobs that have bonuses lower than 2000
SET SERVEROUTPUT ON
DECLARE
no_field exception;
pragma exception_init(no_field, -20101);
BEGIN
UPDATE Sponsors
SET sponsor_id = 0
WHERE company_name LIKE '%Oracle';
IF SQL%NOTFOUND = TRUE THEN
   RAISE_APPLICATION_ERROR(-20101, 'No such sponsor found.');
END IF;
EXCEPTION
   WHEN no_field THEN
      DBMS_OUTPUT.PUT_LINE('No such sponsor found');
END;
/
--create a package
create or replace package player_pack_1 is
--display the number of players in a certain team
function get_no_players(v_team_id number) return number;
```

```
--Estimate the average amount of money a future player can win per month given the job_id
function average_salary(v_job_id number) return number;

--show the date of the match with a given id
function date_of_match(v_match_id number) return date;

--increase the salary of a certain player
procedure increase_salary(v_player_id number, v_percentage number);

--change the contract of 'Sponsor2' to end on the current date
procedure contract_ends(v_name string);
end;

create or replace package body player_pack_1 is

function get_no_players(v_team_id number) return number is
v_number number;
begin
select count(*) into v_number from players where team_id = v_team_id;
return v_number;
end;

function average_salary(v_job_id number) return number is
v_avg_salary number:=0;
begin
```

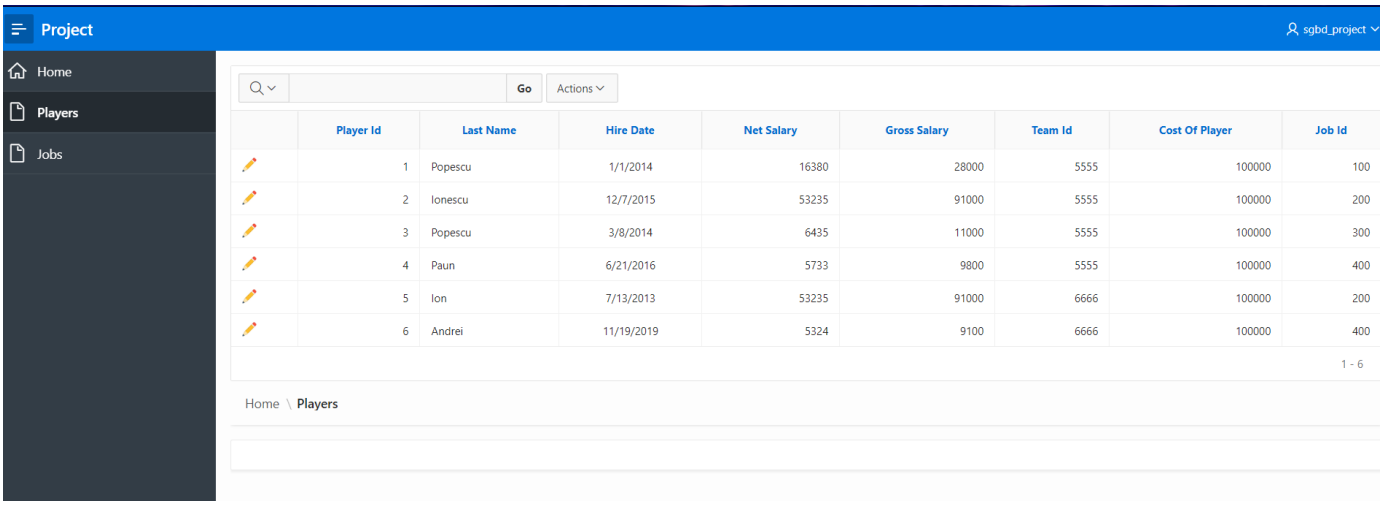
```
select round(((max_salary - min_salary)/2+bonuses)/12) into v_avg_salary
   from jobs where job_id = v_job_id;
return v_avg_salary;
end;

function date_of_match(v_match_id number) return date is
v_date date;
begin
select match_date into v_date from matches where v_match_id = match_id;
return v_date;
end;

procedure increase_salary(v_player_id number, v_percentage number) is
no_player exception;
pragma exception_init(no_player, -20101);
begin
update players
set net_salary = net_salary + net_salary*v_percentage/100
where player_id = v_player_id;
if sql%notfound = true then
   raise_application_error(-2101, 'There was no player found');
end if;
exception when no_player then
   null;
end;
```

```
procedure contract_ends(v_name string) is

v_end date;

begin

select sysdate into v_end from dual;

update sponsors

set contract_end_date = v_end

where company_name = v_name;

end;

end;


set SERVEROUTPUT on;

begin

dbms_output.put_line('The number of players from the requested team is
'||player_pack.get_no_players(5555));

dbms_output.put_line('The average monthly salary for the requested job is
'||player_pack.average_salary(400));

dbms_output.put_line('The date of the requested match is '||player_pack.date_of_match(12));

player_pack.increase_salary(78, 5);

player_pack.increase_salary(3, 5);

player_pack.contract_ends('Sponsor3');

EXCEPTION

   WHEN OTHERS THEN

      dbms_output.put_line('Wrong data.');

end;

/
```

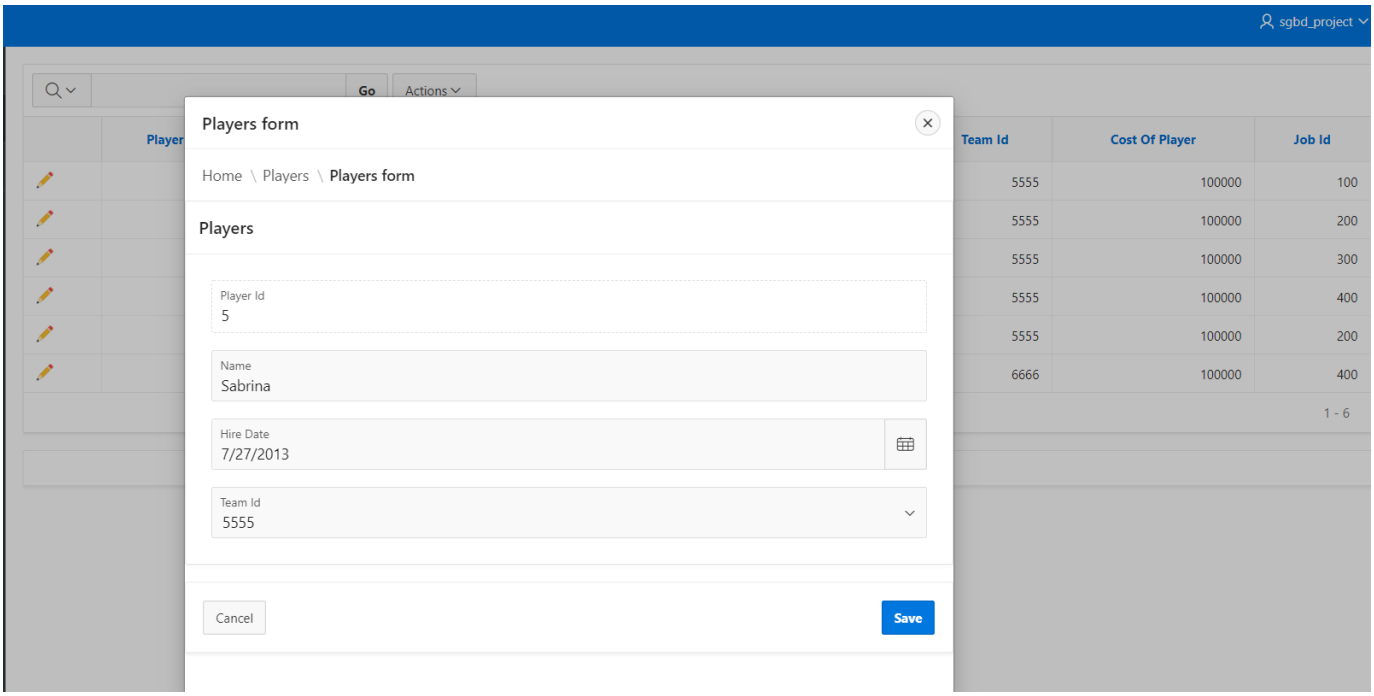
```
--create a trigger that doesn't allow a player to be in a different team than 5555 and 6666

CREATE OR REPLACE TRIGGER non_existent_team BEFORE INSERT ON PLAYERS for each row

BEGIN

   IF :new.team_id not in (5555, 6666) then

      raise_application_error(-20001, 'Wrong team ID');

   end if;

END;

/


--to check if the trigger is triggered

INSERT INTO Players (PLAYER_ID, LAST_NAME, HIRE_DATE, NET_SALARY, GROSS_SALARY, TEAM_ID,
COST_OF_PLAYER, JOB_ID) VALUES (90, 'Andrei', '19-NOV-2019', 9000, 9100, 100, 100000, 400);


--construct a trigger that won't allow you to input a hire_date that is in the future

CREATE OR REPLACE TRIGGER invalid_time BEFORE INSERT OR UPDATE ON PLAYERS for each row

BEGIN

   IF :new.hire_date > sysdate then

      raise_application_error(-20002, 'The date is not valid.');

   end if;

END;

/


--to check

INSERT INTO Players (PLAYER_ID, LAST_NAME, HIRE_DATE, NET_SALARY, GROSS_SALARY, TEAM_ID,
COST_OF_PLAYER, JOB_ID) VALUES (90, 'Andrei', '19-NOV-2021', 9000, 9100, 5555, 100000, 400);
```

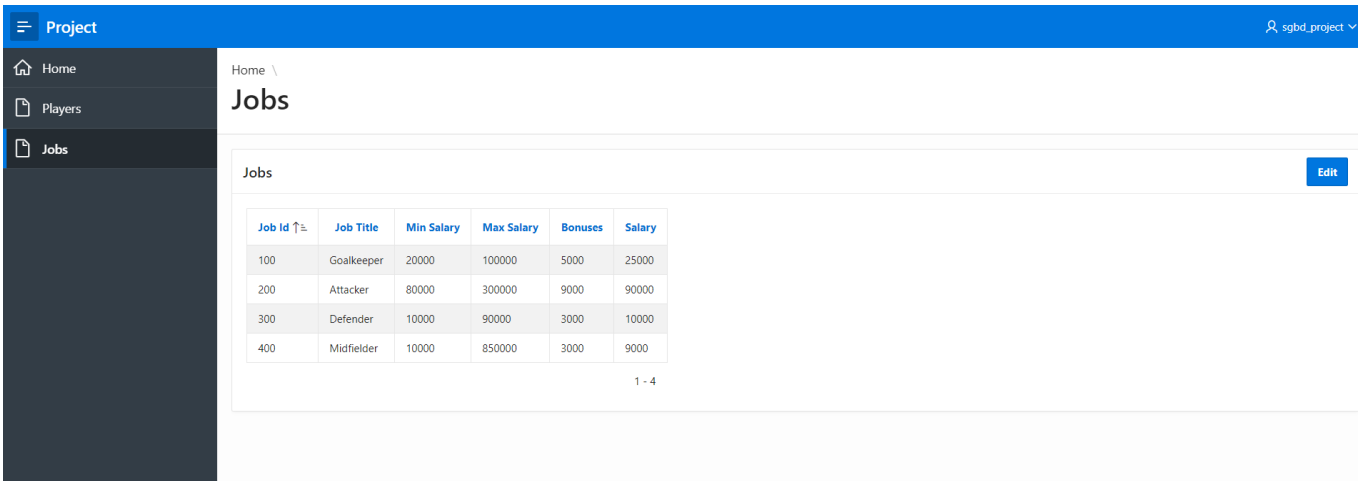
APEX part

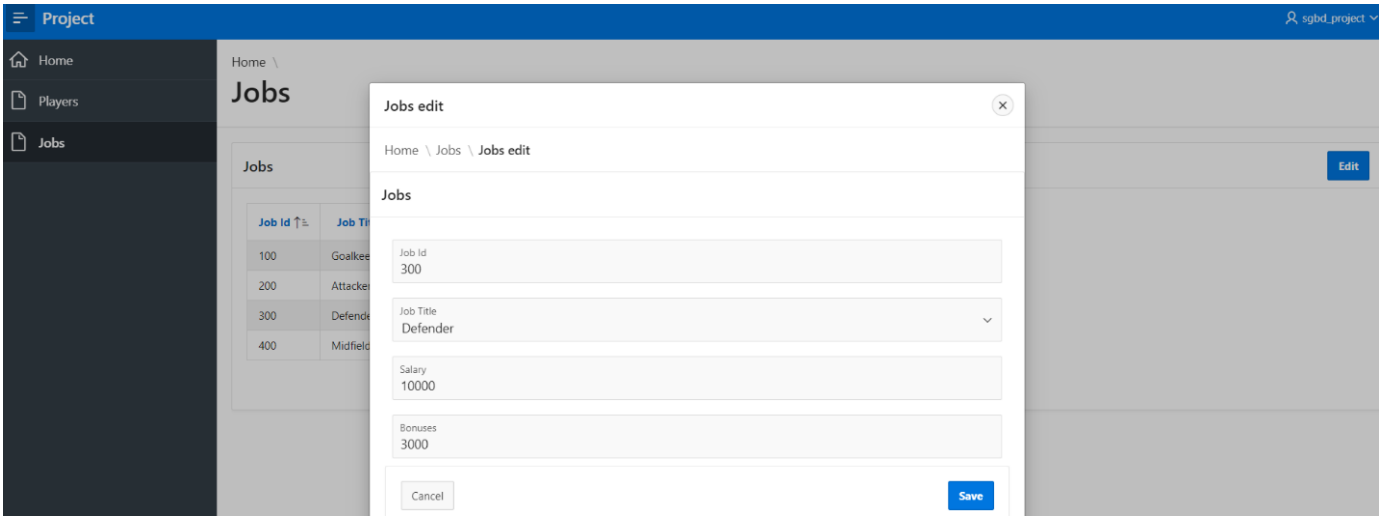
Link: https://rpgzn0h8icu4sy6-sgbd.adb.eu-frankfurt-1.oraclecloudapps.com/ords/f?p=106:1:113874474989329::::



Player Id	Last Name	Hire Date	Net Salary	Gross Salary	Team Id	Cost Of Player	Job Id
1	Popescu	1/1/2014	16380	28000	5555	100000	100
2	Ionescu	12/7/2015	53235	91000	5555	100000	200
3	Popescu	3/8/2014	6435	11000	5555	100000	300
4	Paun	6/21/2016	5733	9800	5555	100000	400
5	Ion	7/13/2013	53235	91000	6666	100000	200
6	Andrei	11/19/2019	5324	9100	6666	100000	400




Project
sgbd_project
Home
Players
Jobs
Home \
Jobs
Jobs
Edit
Job Id
Job Title
Min Salary
Max Salary
Bonuses
Salary
100 Goalkeeper 20000 100000 5000 25000
200 Attacker 80000 300000 9000 90000
300 Defender 10000 90000 3000 10000
400 Midfielder 10000 850000 3000 9000
1 - 4


Project
sgbd_project
Home
Players
Jobs
Home \
Jobs
Jobs edit
Home \ Jobs \ Jobs edit
Jobs
Edit
Job Id
300
Job Title
Defender
Salary
10000
Bonuses
3000
Cancel
Save