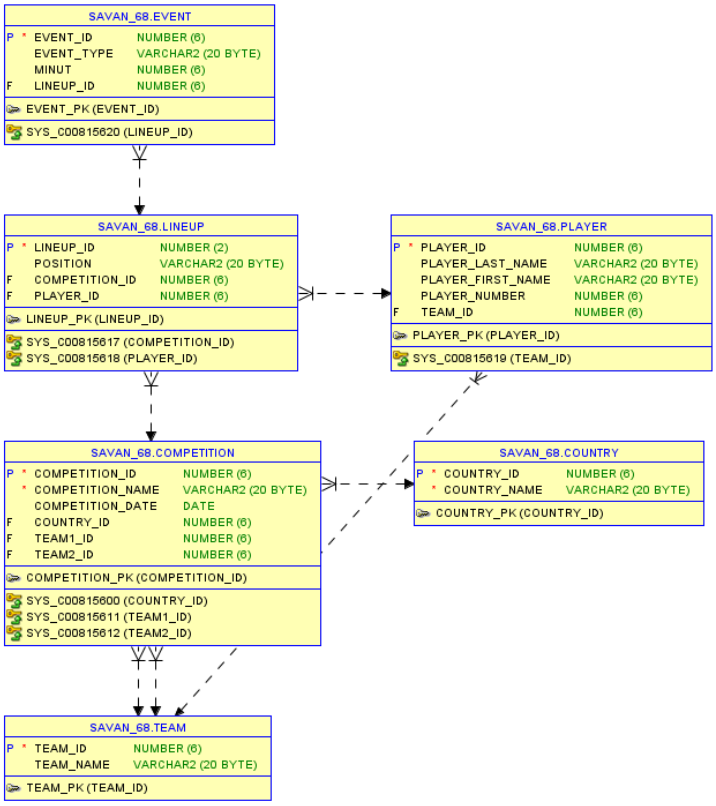
SGBD ORACLE PROJECT

Sava Nicolae

1068

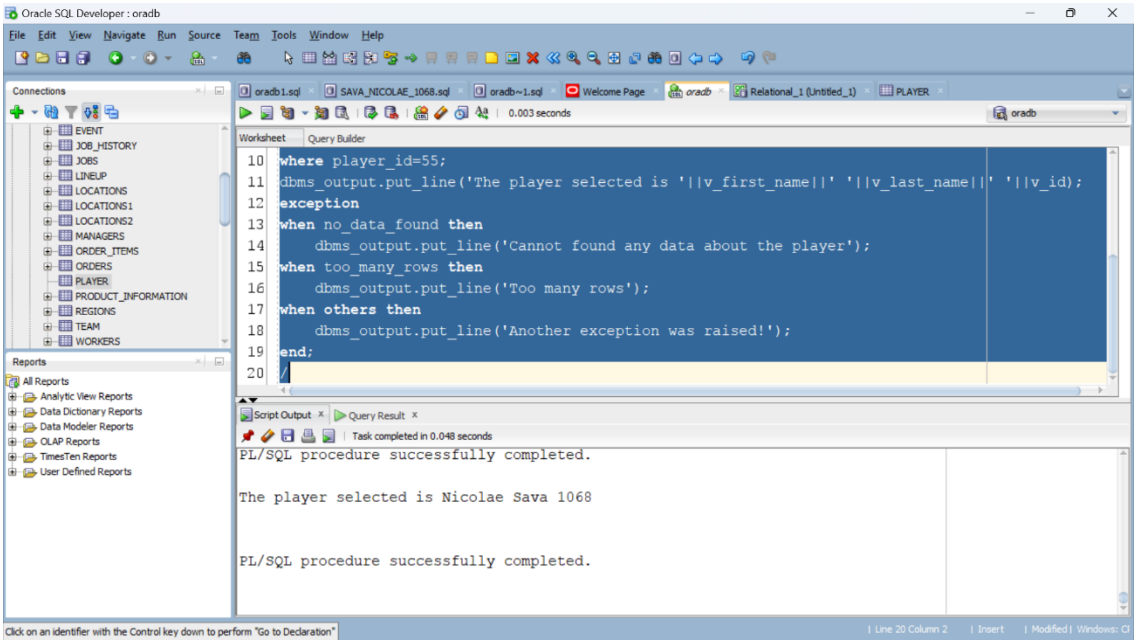
Database schema



I did not change the schema from the last semester.

Problems

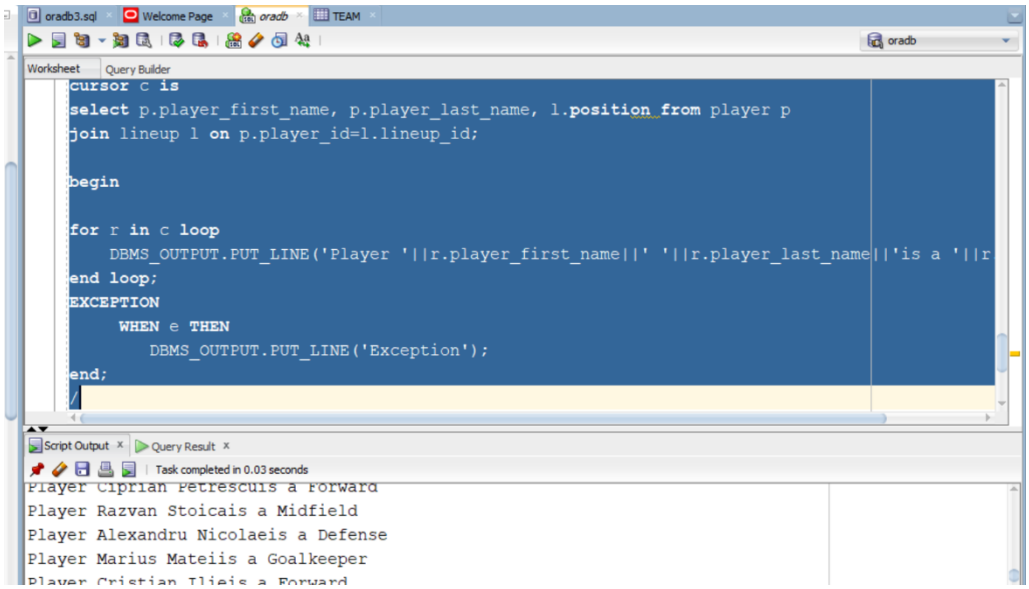
1. Create a declare block that selects the first name, last name and player number from player table and throws 3 implicit exceptions.




```
set serveroutput on

declare
v_first_name varchar2(30);
v_last_name varchar2(30);
v_id number(10);
begin
select player_first_name, player_last_name , player_number into v_first_name,
v_last_name, v_id
from player
where player_id=55;
dbms_output.put_line('The player selected is '||v_first_name||' '||v_last_name||' '||v_id);
exception
when no_data_found then
   dbms_output.put_line('Cannot found any data about the player');
when too_many_rows then
   dbms_output.put_line('Too many rows');
when others then
   dbms_output.put_line('Another exception was raised!');
end;
/
```

2. Create a declare block which uses an explicit cursor to select the full name of the players and shows for each player the position they are playing.

```
declare

e exception;

PRAGMA EXCEPTION_INIT(e,-00955);

cursor c is

select p.player_first_name, p.player_last_name, l.position from player p

join lineup l on p.player_id=l.lineup_id;


begin


for r in c loop

   DBMS_OUTPUT.PUT_LINE('Player '||r.player_first_name||' '||r.player_last_name||'is a '||r.position);

end loop;

EXCEPTION

    WHEN e THEN

      DBMS_OUTPUT.PUT_LINE('Exception');

end;

/
```



3. Create a declare block which will use a cursor to select the lineup id and the competition name and prints out each lineup on competitions.

```
declare
cursor c is
   select l.lineup_id, c.competition_name from competition c
   join lineup l on c.competition_id=l.competition_id;
begin
   for r in c loop
      case
         when r.competition_name='Champions League' then
            dbms_output.put_line('The lineup with id '||r.lineup_id||' is in Champions League');
         when r.competition_name='Premier League' then
            dbms_output.put_line('The lineup with id '||r.lineup_id||' is in Premier League');
         when r.competition_name='La Liga' then
            dbms_output.put_line('The lineup with id '||r.lineup_id||' is in La Liga');
      end case;
   end loop;
end;
/
```

4. Declare a collection of players indexed by table which prints out how many players each team have. If the team has no players, next to the id of the team it will be printed “The team has no players registered”.

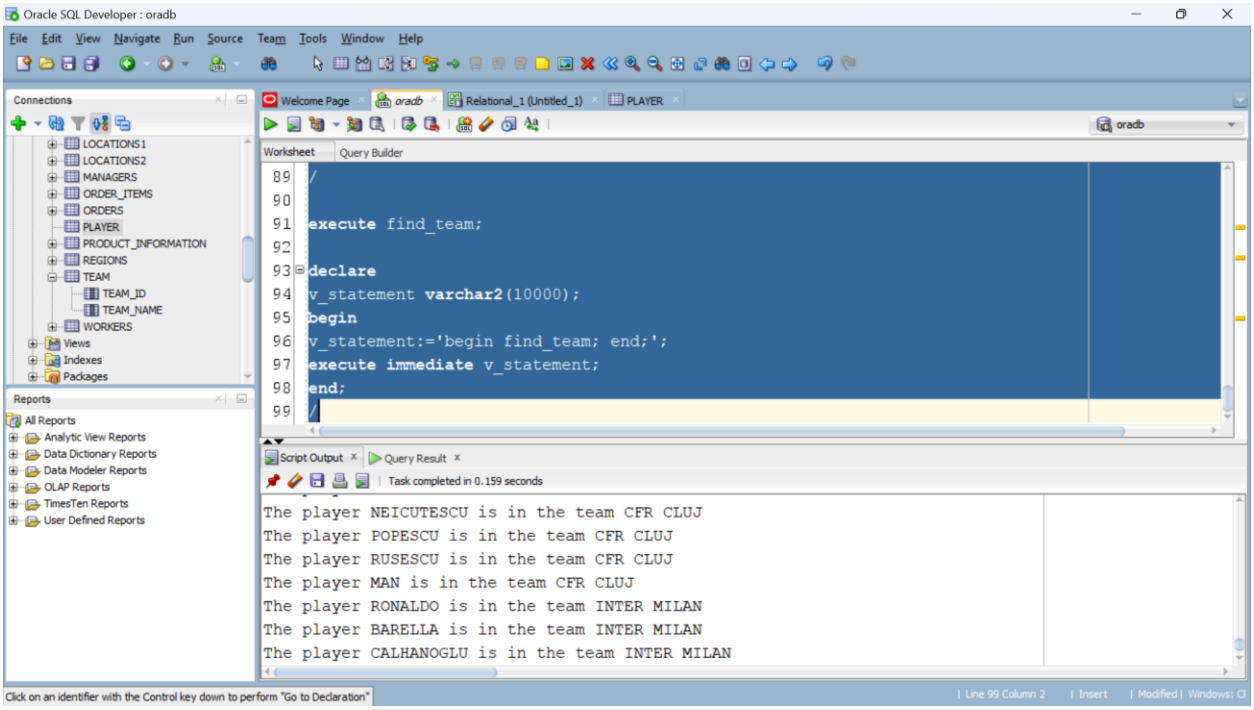
```
declare
  type t_player_names is table of varchar2(150) index by pls_integer;
  t t_player_names;
  i pls_integer;
begin
  SELECT t.team_name||' has '||COUNT(p.player_id)||' players' message  BULK COLLECT INTO t
  FROM player p
```

```
  JOIN team t ON p.team_id=t.team_id
  GROUP BY team_name
  order by COUNT(p.player_id) desc;
  t(-100000000):='The team has no players registered';
  dbms_output.put_line(t.count);
  dbms_output.put_line(t.first);
  dbms_output.put_line(t.last);
  i:=t.first;
  while i is not null loop
     dbms_output.put_line(i||'->'||t(i));
     i:=t.next(i);
  end loop;
  t.delete;
    dbms_output.put_line(t.count);
    dbms_output.put_line(t.first);
    dbms_output.put_line(t.last);
end;
/
```

5. Create a procedure named find_team that will search for the player's last name in the PLAYER table and find the team in which he plays. The procedure will be executed immediate.

```
create or replace procedure find_team is
    cursor c is
        select upper(p.player_last_name) last_name , t.team_name
            from player p
            join team t on p.team_id=t.team_id;
begin
for r in c loop
    dbms_output.put_line('The player '||r.last_name||' is in the team '||r.team_name);
```

```
end loop;


end;
/


execute find_team;


declare
v_statement varchar2(10000);
begin
v_statement:='begin find_team; end;';
execute immediate v_statement;
end;
/
```



6. Create a function which returns the last minute when a penalty was awarded. Call the function in another declare block.

```
create or replace function max_minute_penalty
return number is
v_current_minute number;
v_max_minute number:=0;
cursor c is
   select minut from event
   where event_type='Penalty';
begin
   for r in c loop
      v_current_minute:=r.minut;
      if v_current_minute>v_max_minute then
         v_max_minute:=v_current_minute;
      end if;
   end loop;
   return v_max_minute;
end;
/


declare
val number;
begin
val:=max_minute_penalty();
dbms_output.put_line('The last minute when a penalty was awarded was '||val);
end;
/
```

7. Create a trigger which will show a message after an update on the TEAM table will be called. Call the update to show the trigger.

```
create or replace trigger team_update
after update on team
begin
dbms_output.put_line('A row in the TEAM table has been updated');
end;
/

update team
set team_name='CHELSEA FC'
WHERE TEAM_ID=6;
```

8. Create a procedure which will use a cursor to show each competition result. The procedure will be called GetCompetitionResults.

```
CREATE OR REPLACE PROCEDURE GetCompetitionResults is
cursor c is select c.team1_id,c.team2_id,l.lineup_id,e.event_type, e.minut from competition c
join lineup l on c.competition_id=l.competition_id
join event e on l.lineup_id=e.lineup_id;
begin
for r in c
LOOP
 DBMS_OUTPUT.PUT_LINE('Team 1: ' || r.team1_id);
   DBMS_OUTPUT.PUT_LINE('Team 2: ' || r.team2_id);
   DBMS_OUTPUT.PUT_LINE('Event Type: ' || r.event_type);
   DBMS_OUTPUT.PUT_LINE('Minute: ' || r.minut);
   DBMS_OUTPUT.PUT_LINE('------------------------');
END LOOP;
END;
/
```

```
execute GetCompetitionResults;
```

9. Create a procedure which will get each event details using a cursor. The procedure will throw an explicit exception.

```
create or replace procedure getevent is

e exception;

PRAGMA EXCEPTION_INIT(e,-00955);

cursor c is select event_id,event_type, minut from event;

begin

for r in c

loop

DBMS_OUTPUT.PUT_LINE('Event ID: '||r.event_id);

DBMS_OUTPUT.PUT_LINE('Event Type: '|| r.event_type);

DBMS_OUTPUT.PUT_LINE('Event Minute:'||r.minut);

DBMS_OUTPUT.PUT_LINE('------------------');

end loop;


exception when e then

dbms_output.put_line('Exception');

end;

/

execute getevent;
```

10. Create a function named gettotalevents which will return a number stored in a variable v_total_events. The function will counr the total number of events from EVENT table. Call the function in a declare block.

```
create or replace function gettotalevents

return number is

v_total_events NUMBER;

begin
```

```
select count(*) into v_total_events
from event;
return v_total_events;
end;
/

declare
v_total_events number;
begin
v_total_events:=gettotalevents();
dbms_output.put_line('Number of events'||v_total_events);
end;
/
```

11. Create a function named foul_player which will return a varchar2, the name of the player which receives a foul. Call the function in a declare block.

```
create or replace function foul_player
return varchar2 is
v_foul_player varchar2(100);
begin
select p.player_last_name into v_foul_player from player p
join lineup l on p.player_id=l.lineup_id
join event e on l.lineup_id=e.lineup_id
where e.event_type='Foul';

return v_foul_player;
end;
/
```

```
declare
v_player varchar2(100);
begin
v_player:=foul_player();
dbms_output.put_line('The player who made a foul is '||v_player||'.');
end;
/
```

12. Create a row level trigger which updates the competition_id into the LINEUP table

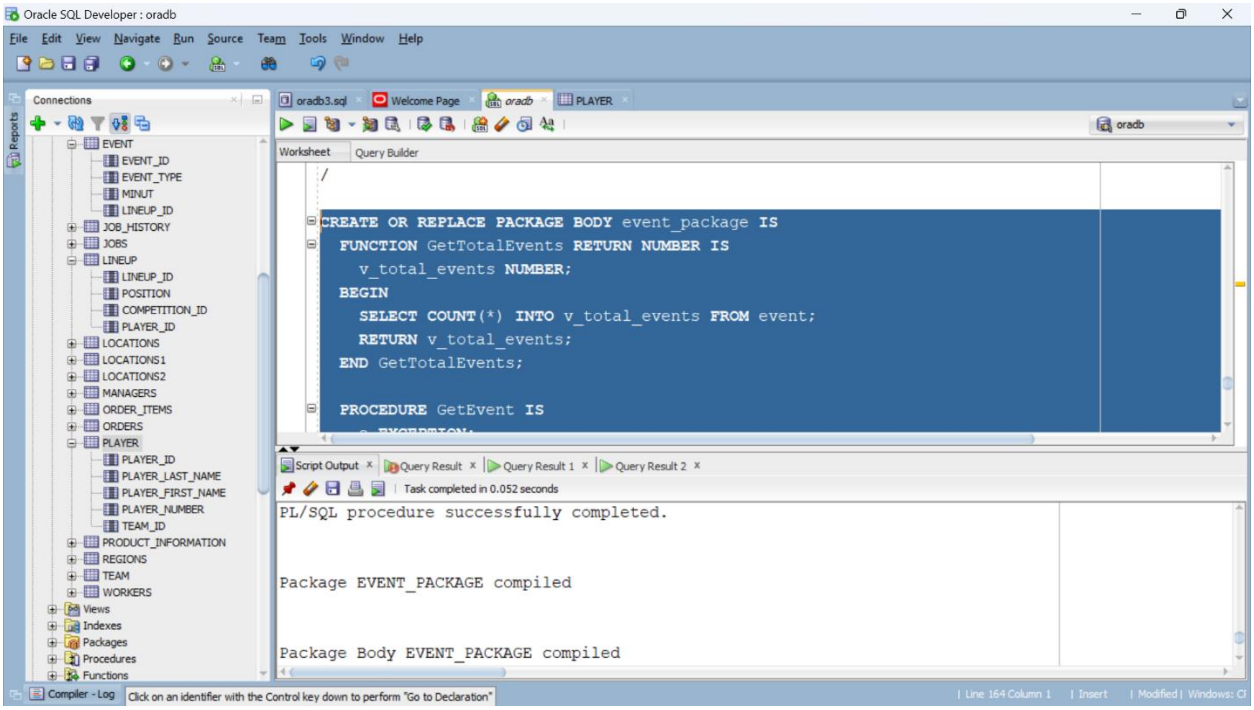
```
CREATE OR REPLACE TRIGGER UpdateCompetitions
AFTER INSERT OR DELETE ON LINEUP
FOR EACH ROW
DECLARE
  v_competition_id COMPETITION.competition_id%TYPE;
BEGIN
  IF INSERTING THEN
    v_competition_id := :new.competition_id;
  ELSIF DELETING THEN
    v_competition_id := :old.competition_id;
  END IF;

  DBMS_OUTPUT.PUT_LINE('Competition ID replaced with ' || v_competition_id || ' updated.
');
END;
/
```

13. Create a declare block which uses an implicit cursor to count the number of rows updated in the team 1 when we increase each player number by 1.

```
DECLARE
  v_num_rows NUMBER;
```

```
BEGIN
  UPDATE player
  SET player_number = player_number+1
  WHERE team_id = 1;

  v_num_rows := SQL%ROWCOUNT;

  DBMS_OUTPUT.PUT_LINE('Number of rows updated: ' || v_num_rows);
END;
/
```

14. Create a package event_package that contains procedure called GetEvents and function called getTotalEvents.




```
CREATE OR REPLACE PACKAGE BODY event_package IS
  FUNCTION GetTotalEvents RETURN NUMBER IS
    v_total_events NUMBER;
```

```
  BEGIN
    SELECT COUNT(*) INTO v_total_events FROM event;
    RETURN v_total_events;
  END GetTotalEvents;

  PROCEDURE GetEvent IS
    e EXCEPTION;
    PRAGMA EXCEPTION_INIT(e, -00955);
    CURSOR c IS SELECT event_id, event_type, minut FROM event;
  BEGIN
    FOR r IN c LOOP
      DBMS_OUTPUT.PUT_LINE('Event ID: ' || r.event_id);
      DBMS_OUTPUT.PUT_LINE('Event Type: ' || r.event_type);
      DBMS_OUTPUT.PUT_LINE('Event Minute: ' || r.minut);
      DBMS_OUTPUT.PUT_LINE('------------------');
    END LOOP;
  EXCEPTION
    WHEN e THEN
      DBMS_OUTPUT.PUT_LINE('Exception');
  END GetEvent;
END event_package;
/
```

15. Create a statement level trigger called DisplatTotalEventsTrigger which is activated after an insert, delete or update on the event table and will calculate the total events in the event table.

```
CREATE OR REPLACE TRIGGER DisplayTotalEventsTrigger
AFTER INSERT OR DELETE OR UPDATE ON EVENT
DECLARE
  v_total_events NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_total_events
  FROM EVENT;
  DBMS_OUTPUT.PUT_LINE('Total events: ' || v_total_events);
END;
/
```

16. Create a row level trigger that will activate when inserting a new row in player table and checks if the row we insert has a play number, and if it does not have, it will be assigned to 0 by the trigger.

```
CREATE OR REPLACE TRIGGER UpdatePlayerNumberTrigger
BEFORE INSERT ON PLAYER
FOR EACH ROW
BEGIN
  IF :new.player_number IS NULL THEN
    :new.player_number := 0;
  END IF;
END;
/
```