Football Network Management System

Database Programming

Project Report

Tycjan Fortuna
Team leader
ID: 242213

Filip Krylecki
ID: 242220

Marek Kopania
ID: 234760

Paulina Klewin
ID: 247028

Information Technology
IFE, 4th semester
Compulsory course

May 29, 2023

Contents

1 Project description **4**

1.1 Problem description 4

1.2 Aims 4

1.3 System features and functions 4

1.4 Entity Relationship Diagram (ERD) 5

2 Implementation **6**

2.1 Scripts for creating the database 6

2.2 Scripts for populating the database 8

2.3 SQL queries 9

2.4 Procedures 16

2.5 Functions 20

2.6 Triggers 25

2.7 Package 28

2.8 Other additional database structures 31

Listings

1 Scripts creating the database (DBTableCreation.sql file) 6

2 Scripts populating the database (DBTableInsertion.sql file) 8

3 SQL queries (DBQueries.sql file) 9

4 Procedures (DBProcedures.sql file) 16

5 Functions (DBFunctions.sql file) 20

6 Triggers (DBTriggers.sql file) 25

7 Package (DBPackage.sql file) 28

8 Other additional structures (DBAdditionalStructures.sql file) 31

1 Project description

1.1 Problem description

Our client wants a management system to keep information about the professional football division of the Spanish football league system, La Liga. They wish to keep and update relevant information, i.e. about the matches, participating football clubs, stadiums and varied tickets. Throughout the season, twenty football teams compete against each other twice, at home and away, resulting in 38 matches per club. The client wants to know which three teams are the best in La Liga. They can display record of currently playing football players as well as detailed information on each individual. The client can search for teams and stadiums by given city. Additionally, it is possible for them to generate a report containing scores of the matches registered in the database. Thanks to our management system the client can see the average remuneration of football players from a selected club. There is a possibility to check how commercially successful any of the matches was. Employees of our client are able to maintain the information about La Liga's progress by adding new data and updating the existing one. They can also parse XML report. Our client requested verification of the data that is entered into the system.

1.2 Aims

The aim of the management system is to maintain the information for the client's purposes, including its storage and display. The management system enables maintenance of large amounts of data, allowing further expansion with data from following seasons of La Liga. The system must fulfill the requests of the client as well as verify correctness of the input data. The up-to-date information is displayed in a manner requested by the client.

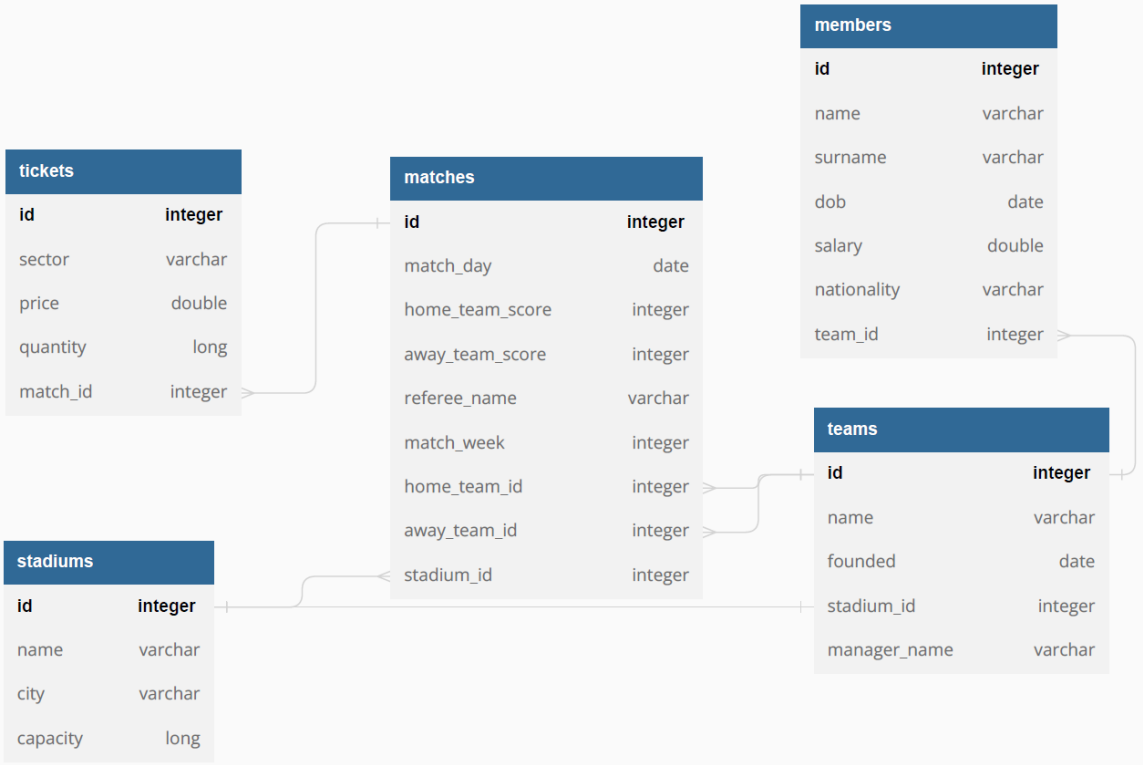
1.3 System features and functions

The football network management system allows:

- alteration and storage of data concerning matches
- alteration and storage of data concerning football teams and their members
- alteration and storage of data concerning stadiums

- alteration and storage of data concerning tickets sold for the matches
- verification of input data

1.4 Entity Relationship Diagram (ERD)



2 Implementation

2.1 Scripts for creating the database

```
-- Create Stadiums table
CREATE TABLE stadiums (
  id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,
  name VARCHAR2(255),
  city VARCHAR2(255),
  capacity NUMBER,
  PRIMARY KEY (id)
);

-- Create Teams table
CREATE TABLE teams (
  id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,
  name VARCHAR2(255),
  founded DATE,
  stadium_id NUMBER,
  manager_name VARCHAR2(255),
  PRIMARY KEY (id),
  FOREIGN KEY (stadium_id) REFERENCES stadiums(id)
);

-- Create Matches table
CREATE TABLE matches (
  id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,
  match_date DATE,
  home_team_score NUMBER,
  away_team_score NUMBER,
  referee_name VARCHAR2(255),
  match_week NUMBER,
  home_team_id NUMBER,
  away_team_id NUMBER,
```

```
  stadium_id NUMBER,
  PRIMARY KEY (id),
  FOREIGN KEY (stadium_id) REFERENCES stadiums(id),
  FOREIGN KEY (home_team_id) REFERENCES teams(id),
  FOREIGN KEY (away_team_id) REFERENCES teams(id)
);

-- Create Tickets table
CREATE TABLE tickets (
  id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,
  sector VARCHAR2(255),
  price NUMBER,
  quantity NUMBER,
  match_id NUMBER,
  PRIMARY KEY (id),
  FOREIGN KEY (match_id) REFERENCES matches(id)
);

-- Create Members table
CREATE TABLE members (
  id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY,
  name VARCHAR2(255),
  surname VARCHAR2(255),
  dob DATE,
  salary NUMBER,
  nationality VARCHAR2(255),
  team_id NUMBER,
  PRIMARY KEY (id),
  FOREIGN KEY (team_id) REFERENCES teams(id)
);
```

Listing 1: Scripts creating the database (`DBTableCreation.sql` file)

2.2 Scripts for populating the database

```
-- Populate Stadiums table
INSERT INTO stadiums VALUES (1,'Camp Nou','Barcelona',99354);
INSERT INTO stadiums VALUES (2,'Estadio Metropolitano','Madrid',68456);
(...)
INSERT INTO stadiums VALUES (9,'Campo de Futbol de Vallecas','Puente de Vallecas',14708);
INSERT INTO stadiums VALUES (10,'El Sadar','Pamplona',23576);

-- Populate Teams table
INSERT INTO teams VALUES (1,'Barcelona',TO_DATE('1899-11-29', 'YYYY-MM-DD'),1,'Xavier Hernandez Creus');
INSERT INTO teams VALUES (2,'Atletico Madrid',TO_DATE('1903-04-26', 'YYYY-MM-DD'),2,'Diego Simeone');
(...)
INSERT INTO teams VALUES (9,'Rayo Vallecano de Madrid',TO_DATE('1919-03-18', 'YYYY-MM-DD'),9,'Andoni Iraola');
INSERT INTO teams VALUES (10,'Osasuna',TO_DATE('1890-01-25', 'YYYY-MM-DD'),10,'Jagoba Arrasate');

-- Populate Matches table
INSERT INTO matches VALUES (1, TO_DATE('2023-08-05', 'YYYY-MM-DD'), 3, 2, 'Antonio Mateu Lahoz', 1, 1, 2, 1);
INSERT INTO matches VALUES (2, TO_DATE('2023-08-05', 'YYYY-MM-DD'), 1, 0, 'Carlos Del Cerro Grande', 1, 3, 4, 3);
(...)
INSERT INTO matches VALUES (249, TO_DATE('2025-04-20', 'YYYY-MM-DD'), 1, 0, 'Carlos Del Cerro Grande', 3, 6, 1, 6);
INSERT INTO matches VALUES (250, TO_DATE('2025-04-26', 'YYYY-MM-DD'), 1, 1, 'Aliyar Aghayev', 4, 7, 10, 7);

-- Populate Tickets table
INSERT INTO tickets VALUES (1, 'A', 50.00, 10000, 1);
```

```
24 INSERT INTO tickets VALUES (2, 'B', 40.00, 15000, 1);
25 (...)
26 INSERT INTO tickets VALUES (979, 'B', 1255.00, 9670, 250);
27 INSERT INTO tickets VALUES (980, 'A', 1245.00, 9680, 250);
28
29 -- Populate Members table
30 INSERT INTO members VALUES (1,'Marc-Andre','ter Stegen',TO_DATE('1992-04-30
       ', 'YYYY-MM-DD'),8900.99,'German',1);
31 INSERT INTO members VALUES (2,'Ronald','Araujo',TO_DATE('1999-03-07', 'YYYY
       -MM-DD'),7858.09,'Uruguayan',1);
32 (...)
33 INSERT INTO members VALUES (107,'Lucas','Torro',TO_DATE('1994-01-01', 'YYYY
       -MM-DD'),9876.00,'Spanish',10);
34 INSERT INTO members VALUES (108,'Jony','Rodriguez',TO_DATE('1991-01-01', '
       YYYY-MM-DD'),9876.00,'Spanish',10);
```

Listing 2: Scripts populating the database (`DBTableInsertion.sql` file)

2.3 SQL queries

```
1 -- Basic queries for our database
2 SELECT * FROM stadiums;
3
4 SELECT * FROM teams;
5
6 SELECT * FROM matches;
7
8 SELECT * FROM tickets;
9
10 SELECT * FROM members;
11
12 -- 1) Retrieve the stadium details for a specific team:
13 SELECT stadiums.* FROM stadiums
```

```
14      JOIN teams ON stadiums.id = teams.stadium_id
15  WHERE teams.name = 'Barcelona';
16
```

ID	NAME	CITY	CAPACITY
1	Camp Nou	Barcelona	20000

Figure 1: Query 1 output

```
17
18  -- 2) Retrieve the matches played in a specific stadium:
19  SELECT * FROM matches
20  WHERE stadium_id = 1;
21
```

ID	MATCH_DATE	HOME_TEAM_SCORE	AWAY_TEAM_SCORE	REFEREE_NAME	MATCH_WEEK	HOME_TEAM_ID	AWAY_TEAM_ID	STADIUM_ID
134	24/07/06	2	1	Jose Maria Sanchez Martinez	5	1	6	1
144	24/07/28	1	0	Vitor Melo Pereira	8	1	6	1
154	24/08/24	1	1	Aliyar Aghayev	12	1	6	1
164	24/09/21	2	1	Jose Maria Sanchez Martinez	15	1	6	1
174	24/10/20	1	0	Vitor Melo Pereira	18	1	6	1

Figure 2: Query 2 output

```
22
23  -- 3) Retrieve the teams playing in a specific match:
24  SELECT home_teams.name AS home_team,
25         away_teams.name AS away_team
26  FROM matches
27      JOIN teams home_teams ON home_teams.id = matches.home_team_id
28      JOIN teams away_teams ON away_teams.id = matches.away_team_id
29  WHERE matches.id = 11;
30
```

HOME_TEAM	AWAY_TEAM
Real Sociedad	Rayo Vallecano de Madrid

Figure 3: Query 3 output

```sql
-- 4) Retrieve all tickets for a specific match:
SELECT * FROM tickets
WHERE match_id = 1;

```

ID	SECTOR	PRICE	QUANTITY	MATCH_ID
1	A	50	10000	1
2	B	40	15000	1
3	C	30	20000	1

Figure 4: `Query 4` output

```sql

-- 5) Retrieve the members of a specific team:
SELECT * FROM members
WHERE team_id = 5;

```

ID	NAME	SURNAME	DOB	SALARY	NATIONALITY	TEAM_ID
45	Sergio	Asenjo	89/06/28	8993	Spanish	5
46	Mario	Gaspar	90/11/24	6000	Spanish	5
47	Pau	Torres	97/01/16	7000	Spanish	5
48	Raul	Albiol	85/09/04	7000	Spanish	5

Figure 5: `Query 5` output

```sql

-- 6) Retrieve the average ticket price for each match:
SELECT matches.id,
       AVG(tickets.price) AS average_ticket_price
FROM matches
    JOIN tickets ON matches.id = tickets.match_id
GROUP BY matches.id;

```

ID	AVERAGE_TICKET_PRICE
107	545
108	550
113	575
124	630
125	635

Figure 6: Query 6 output

```sql
-- 7) Retrieve the team with the highest average score in home matches:
SELECT teams.name,
       AVG(matches.home_team_score) AS average_home_score
FROM teams
    JOIN matches ON teams.id = matches.home_team_id
GROUP BY teams.name
ORDER BY average_home_score DESC
FETCH FIRST ROW ONLY;
```

NAME	AVERAGE_HOME_SCORE
Villarreal	1,46153846153846153846153846153846153846

Figure 7: Query 7 output

```sql
-- 8) Retrieve the matches where the home team scored more than the away
    team:
SELECT matches.* FROM matches
WHERE home_team_score > away_team_score;
```

ID	MATCH_DATE	HOME_TEAM_SCORE	AWAY_TEAM_SCORE	REFEREE_NAME	MATCH_WEEK	HOME_TEAM_ID	AWAY_TEAM_ID	STADIUM_ID
131	24/06/29	2	1	Szymon Marciniak	4	8	9	8
132	24/06/30	1	0	Vitor Melo Pereira	4	9	8	9
134	24/07/06	2	1	Jose Maria Sanchez Martinez	5	1	6	1
135	24/07/07	1	0	Carlos Del Cerro Grande	5	2	5	2

Figure 8: Query 8 output

```sql
-- 9) Retrieve the top 5 stadiums with the highest average of score
SELECT stadiums.name,
       AVG(matches.home_team_score + matches.away_team_score) AS
    average_score
FROM stadiums
    JOIN matches ON stadiums.id = matches.stadium_id
GROUP BY stadiums.name
ORDER BY average_score DESC
FETCH FIRST 5 ROWS ONLY;

```

NAME	AVERAGE_SCORE
Camp Nou	2,22222222222222222222222222222222222222
Estadio de la Ceramica	2,19230769230769230769230769230769230769
Estadio Metropolitano	2,15384615384615384615384615384615384615
Anoeta Stadium	2,04
Estadi Montilivi	2,04

Figure 9: Query 9 output

```sql

-- 10) Retrieve the matches where the ticket quantity sold is greater than
    the average ticket quantity sold for all matches:
SELECT matches.* FROM matches
    JOIN tickets ON matches.id = tickets.match_id
WHERE tickets.quantity > (SELECT AVG(quantity) FROM tickets);

```

ID	MATCH_DATE	HOME_TEAM_SCORE	AWAY_TEAM_SCORE	REFEREE_NAME	MATCH_WEEK	HOME_TEAM_ID	AWAY_TEAM_ID	STADIUM_ID
146	24/08/03	2	1	Jose Maria Sanchez Martinez	9	3	4	3
146	24/08/03	2	1	Jose Maria Sanchez Martinez	9	3	4	3
146	24/08/03	2	1	Jose Maria Sanchez Martinez	9	3	4	3
147	24/08/04	1	0	Carlos Del Cerro Grande	9	4	3	4

Figure 10: Query 10 output

```sql

-- 11) Retrieve the teams with the highest number of members:
SELECT teams.name,
       COUNT(members.id) AS members_count
FROM teams
```

```
83      JOIN members ON teams.id = members.team_id
84  GROUP BY teams.name
85  ORDER BY members_count DESC;
86
```

NAME	MEMBERS_COUNT
Real Betis	11
Real Madrid	11
Villarreal	11
Rayo Vallecano de Madrid	11
Girona	11

Figure 11: Query 11 output

```
87
88  -- 12) Retrieve the team with the highest average member salary:
89  SELECT teams.name,
90         AVG(members.salary) AS average_salary
91  FROM teams
92      JOIN members ON teams.id = members.team_id
93  GROUP BY teams.name
94  ORDER BY average_salary DESC
95  FETCH FIRST ROW ONLY;
96
```

NAME	AVERAGE_SALARY
Osasuna	8917,66666666666666666666666666666667

Figure 12: Query 12 output

```
97
98  -- 13) Retrieve the matches where both home and away teams scored more than
        the average score in all matches:
99  SELECT matches.* FROM matches
100 WHERE home_team_score > (SELECT AVG(home_team_score) FROM matches)
101     AND away_team_score > (SELECT AVG(away_team_score) FROM matches);
102
```

ID	MATCH_...	HOME_TEAM_SCORE	AWAY_TEAM_SCORE	REFEREE_NAME	MATCH_WEEK	HOME_TEAM_ID	AWAY_TEAM_ID	STADIUM_ID
131	24/06/29	2	1	Szymon Marciniak	4	8	9	8
134	24/07/06	2	1	Jose Maria Sanchez Martinez	5	1	6	1
137	24/07/13	2	1	Szymon Marciniak	6	4	3	4
140	24/07/20	2	1	Jose Maria Sanchez Martinez	7	7	10	7
143	24/07/27	2	1	Szymon Marciniak	8	10	7	10

Figure 13: Query 13 output

```
103
104 -- 14) Retrieve the members who have a salary higher than the average
        salary of all members:
105 SELECT * FROM members
106 WHERE salary > (SELECT AVG(salary) FROM members);
107
```

ID	NAME	SURNAME	DOB	SALARY	NATIONALITY	TEAM_ID
1	Marc-Andre	ter Stegen	92/04/30	9791,089	German	1
2	Ronald	Araujo	99/03/07	8643,899	Uruguayan	1
3	Jules	Kounde	98/11/12	7596,578	French	1
4	Andreas	Christensen	96/04/10	8238,23	Danish	1
6	Sergi	Roberto	92/02/07	7803,829	Spanish	1

Figure 14: Query 14 output

```
108
109 -- 15) Retrieve the stadiums that have hosted matches with a match week
        greater than 10:
110 SELECT UNIQUE stadiums.* FROM stadiums
111     JOIN matches ON stadiums.id = matches.stadium_id
112 WHERE matches.match_week > 10;
113
```

ID	NAME	CITY	CAPACITY
7	Estadi Montilivi	Girona	11810
6	Estadio Benito Villamarin	Seville	60720
3	Santiago Bernabeu	Madrid	81044
9	Campo de Futbol de Vallecas	Puente de Vallecas	14708
1	Camp Nou	Barcelona	20000

Figure 15: Query 15 output

```sql
-- 16) Retrieve the teams and their total ticket sales (quantity * price)
    for a specific match:
SELECT teams.name                              AS team_name,
       SUM(tickets.quantity * tickets.price) AS total_sales
FROM teams
    JOIN matches ON teams.id = matches.home_team_id
                    OR teams.id = matches.away_team_id
    JOIN tickets ON matches.id = tickets.match_id
WHERE matches.id = 2
GROUP BY teams.name;

```

TEAM_NAME	TOTAL_SALES
Real Madrid	1470000
Real Sociedad	1470000

Figure 16: `Query 16` output

Listing 3: SQL queries (`DBQueries.sql` file)

2.4 Procedures

```sql
-- Update salary of players of given football club by certain percentage
CREATE OR REPLACE PROCEDURE update_salary (
    p_team_id             NUMBER,
    p_percentage_increase NUMBER
) IS
    CURSOR member_cur IS
    SELECT id,
           salary
    FROM members
    WHERE team_id = p_team_id;

    v_member_id  NUMBER;
```

```
13      v_new_salary NUMBER;
14  BEGIN
15      OPEN member_cur;
16
17      LOOP
18          FETCH member_cur INTO v_member_id,
19                                v_new_salary;
20          EXIT WHEN member_cur%notfound;
21
22          v_new_salary := v_new_salary + ( v_new_salary *
        p_percentage_increase / 100 );
23
24          UPDATE members
25          SET salary = v_new_salary
26          WHERE id = v_member_id;
27      END LOOP;
28
29      CLOSE member_cur;
30      COMMIT;
31  EXCEPTION
32      WHEN OTHERS THEN
33          dbms_output.put_line('Error while increasing salary!');
34  END;
35
36  -- Example of update_salary usage
37  -- BEGIN
38  --      update_salary(1, 10);
39  -- END;
40
41
42  -- Retrieving data about the football players
43  CREATE OR REPLACE PROCEDURE display_member_team IS
44      v_name    members.name%TYPE;
45      v_surname members.surname%TYPE;
```

```
46      v_team    teams.name%TYPE;
47      CURSOR member_cursor IS
48      SELECT m.name,
49             m.surname,
50             t.name
51      FROM members m
52          LEFT JOIN teams   t ON m.team_id = t.id;
53
54  BEGIN
55      OPEN member_cursor;
56
57      LOOP
58          FETCH member_cursor INTO v_name,
59                                   v_surname,
60                                   v_team;
61          EXIT WHEN member_cursor%notfound;
62          dbms_output.put_line(v_name
63                               || ' '
64                               || v_surname
65                               || ' plays for '
66                               || v_team);
67      END LOOP;
68
69      CLOSE member_cursor;
70  END;
71
72  -- Example of display_member_team usage
73  -- BEGIN
74  --     display_member_team;
75  -- END;
76
```

```
Inaki Williams plays for Athletic Club
Raul Garcia plays for Athletic Club
Stole Dimitrievski plays for Rayo Vallecano de Madrid
Sergio Akieme plays for Rayo Vallecano de Madrid
Sergio Gomez plays for Rayo Vallecano de Madrid
```

Figure 17: Procedure `display_member_team` output

```
77
78
79 -- Retrieve data about which teams and stadiums are in a specific city
80 CREATE OR REPLACE PROCEDURE get_teams_and_stadiums_by_city (
81     p_city VARCHAR2
82 ) IS
83     v_team_name    VARCHAR2(255);
84     v_stadium_name VARCHAR2(255);
85 BEGIN
86     SELECT t.name,
87            s.name
88     INTO v_team_name,
89          v_stadium_name
90     FROM teams t
91         JOIN stadiums s ON t.stadium_id = s.id
92     WHERE s.city = p_city;
93
94     dbms_output.put_line('Team: '
95                          || v_team_name
96                          || ', Stadium: '
97                          || v_stadium_name);
98 EXCEPTION
99     WHEN no_data_found THEN
100         dbms_output.put_line('No teams found for the specified city.');
101     WHEN OTHERS THEN
102         dbms_output.put_line('Error while fetching data: ' || sqlerrm);
103 END;
```

```
-- Example of get_teams_and_stadiums_by_city usage
-- DECLARE
--     v_team_name    VARCHAR2(255);
--     v_stadium_name VARCHAR2(255);
-- BEGIN
--     get_teams_and_stadiums_by_city('Pamplona');
-- END;

```

```
Team: Osasuna, Stadium: El Sadar
```

Figure 18: Procedure `get_teams_and_stadiums_by_city` output

Listing 4: Procedures (`DBProcedures.sql` file)

2.5 Functions

```
-- Calculate average salary of a specific football club
CREATE OR REPLACE FUNCTION calculate_average_salary_by_team (
    p_team_id NUMBER
) RETURN NUMBER AS
    v_total_salary   NUMBER := 0;
    v_member_count   NUMBER := 0;
    v_average_salary NUMBER := 0;
    no_members_exception EXCEPTION;
BEGIN
    FOR member_rec IN (SELECT salary
                       FROM members
                       WHERE team_id = p_team_id
    ) LOOP
        v_total_salary := v_total_salary + member_rec.salary;
        v_member_count := v_member_count + 1;
    END LOOP;
```

```
17
18      IF v_member_count > 0 THEN
19          v_average_salary := v_total_salary / v_member_count;
20      ELSE
21          RAISE no_members_exception;
22      END IF;
23
24      RETURN v_average_salary;
25  EXCEPTION
26      WHEN no_members_exception THEN
27          dbms_output.put_line('Error: ' || sqlerrm);
28          RETURN NULL;
29  END;
30  /
31
32  -- Example of calculate_average_salary_by_team usage
33  -- DECLARE
34  --      v_avg_salary NUMBER;
35  -- BEGIN
36  --      v_avg_salary := calculate_average_salary_by_team(1);
37  --      dbms_output.put_line('Average Salary: ' || v_avg_salary);
38  -- END;
39
```

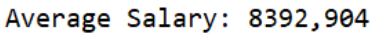


Figure 19: Function `calculate_average_salary_by_team` output

```
40
41
42  -- Calculate total revenue of a specific match
43  CREATE OR REPLACE FUNCTION calculate_total_revenue (
44      p_match_id NUMBER
45  ) RETURN NUMBER AS
46      v_total_revenue NUMBER := 0;
```

```
47      v_total_salary  NUMBER := 0;
48      v_net_revenue   NUMBER := 0;
49  BEGIN
50      SELECT SUM(price * quantity)
51      INTO v_total_revenue
52      FROM tickets
53      WHERE match_id = p_match_id;
54
55      SELECT SUM(salary)
56      INTO v_total_salary
57      FROM members
58      WHERE team_id IN (SELECT home_team_id
59                        FROM matches
60                        WHERE id = p_match_id
61                        UNION
62                        SELECT away_team_id
63                        FROM matches
64                        WHERE id = p_match_id);
65
66      v_net_revenue := v_total_revenue - v_total_salary;
67      RETURN v_net_revenue;
68  END;
69  /
70
71  -- Example of calculate_total_revenue usage
72  -- DECLARE
73  --     revenue NUMBER;
74  -- BEGIN
75  --     revenue := calculate_total_revenue(1);
76  --     dbms_output.put_line('Revenue: ' || revenue);
77  -- END;
78
```

```
Revenue: 1529370,856
```

Figure 20: Function `calculate_total_revenue` output

```
79 -- Retrieve data about top three football clubs
80 CREATE OR REPLACE FUNCTION get_top_teams RETURN SYS_REFCURSOR AS
81     v_team_cur SYS_REFCURSOR;
82 BEGIN
83     OPEN v_team_cur FOR WITH team_points AS (
84         SELECT t.id,
85                t.name,
86                SUM(CASE
87                        WHEN m.home_team_id = t.id THEN
88                            CASE
89                                WHEN m.home_team_score > m.away_team_score
90                                    THEN 3
91                                WHEN m.home_team_score = m.away_team_score
92                                    THEN 1
93                                ELSE
94                                    0
95                            END
96                        ELSE
97                            CASE
98                                WHEN m.away_team_score > m.home_team_score
99                                    THEN 3
100                               WHEN m.away_team_score = m.home_team_score
101                                   THEN 1
102                               ELSE
103                                   0
104                           END
105                   END) AS total_points
106        FROM teams t
107            LEFT JOIN matches m ON m.home_team_id = t.id
108                                   OR m.away_team_id = t.id
```

```
109         GROUP BY t.id,
110                  t.name)
111                         SELECT id,
112                                name,
113                                total_points
114                         FROM (SELECT id,
115                                      name,
116                                      total_points,
117                                      ROW_NUMBER()
118                                      OVER(ORDER BY total_points DESC) AS rn
119                               FROM team_points)
120                         WHERE rn <= 3;
121
122     RETURN v_team_cur;
123 END;
124 /
125
126 -- Example of get_top_teams usage
127 -- DECLARE
128 --     v_teams        SYS_REFCURSOR;
129 --     v_team_id      NUMBER;
130 --     v_team_name    VARCHAR2(255);
131 --     v_total_points NUMBER;
132 -- BEGIN
133 --     v_teams := get_top_teams();
134 --     LOOP
135 --         FETCH v_teams INTO v_team_id,
136 --                            v_team_name,
137 --                            v_total_points;
138 --         EXIT WHEN v_teams%notfound;
139 --         dbms_output.put_line('Team: '
140 --                              || v_team_name
141 --                              || ', Total Points: '
142 --                              || v_total_points);
```

```
143 --      END LOOP;
144
145 --      CLOSE v_teams;
146 -- END;
147
```

```
Team: Villarreal, Total Points: 73
Team: Atletico Madrid, Total Points: 72
Team: Barcelona, Total Points: 67
```

Figure 21: Function `get_top_teams` output

Listing 5: Functions (`DBFunctions.sql` file)

2.6 Triggers

```
1  -- Trigger for the Stadiums table to enforce a minimum capacity
2  CREATE OR REPLACE TRIGGER check_if_stadium_capacity_is_big_enough BEFORE
3      INSERT OR UPDATE ON stadiums
4      FOR EACH ROW
5  DECLARE
6      capacity_too_small EXCEPTION;
7  BEGIN
8      IF :new.capacity < 1000 THEN
9          RAISE capacity_too_small;
10     END IF;
11 EXCEPTION
12     WHEN capacity_too_small THEN
13         raise_application_error(-20001, 'Stadium capacity must be at least
       1000.');
14 END;
15
16 -- Example to check the trigger
17 -- INSERT INTO stadiums VALUES (1111,'El Sadar','Pamplona',100);
```

```
18 -- Trigger for the Matches table to validate scores
19 CREATE OR REPLACE TRIGGER check_if_match_score_is_correct BEFORE
20     INSERT OR UPDATE ON matches
21     FOR EACH ROW
22 BEGIN
23     IF :new.home_team_score < 0 OR :new.away_team_score < 0 THEN
24         raise_application_error(-20002, 'Invalid match scores.');
25     END IF;
26 END;
27
28 -- Example to check the trigger
29 -- INSERT INTO matches VALUES (1111, TO_DATE('2023-08-05', 'YYYY-MM-DD'),
       -3, 2, 'Antonio Mateu Lahoz', 1, 1, 2, 1);
30
31
32 -- Trigger for the Matches table to validate match date
33 CREATE OR REPLACE TRIGGER check_if_match_date_is_valid BEFORE
34     INSERT OR UPDATE ON matches
35     FOR EACH ROW
36 BEGIN
37     IF :new.match_date < sysdate THEN
38         raise_application_error(-20003, 'Match date cannot be in the past.'
       );
39     END IF;
40 END;
41
42 -- Example to check the trigger
43 -- INSERT INTO matches VALUES (11111, TO_DATE('2021-08-05', 'YYYY-MM-DD'),
       3, 2, 'Antonio Mateu Lahoz', 1, 1, 2, 1);
44
45
46 -- Trigger for the Matches table to validate match week
47 CREATE OR REPLACE TRIGGER check_if_match_week_is_valid BEFORE
48     INSERT OR UPDATE ON matches
```

```
    FOR EACH ROW
BEGIN
    IF :new.match_week < 1 OR :new.match_week > 48 THEN
        raise_application_error(-20004, 'Match week must be between 1 and 38.');
    END IF;
END;

-- Example to check the trigger
-- INSERT INTO matches VALUES (23228, TO_DATE('2025-03-29', 'YYYY-MM-DD'), 1, 1, 'Aliyar Aghayev', 410, 5, 2, 5);


-- Trigger for the Matches table to validate home and away teams
CREATE OR REPLACE TRIGGER check_if_home_and_away_teams_are_different BEFORE
    INSERT OR UPDATE ON matches
    FOR EACH ROW
BEGIN
    IF :new.home_team_id = :new.away_team_id THEN
        raise_application_error(-20005, 'Home and away teams must be different.');
    END IF;
END;

-- Example to check the trigger
-- INSERT INTO matches VALUES (238, TO_DATE('2025-03-29', 'YYYY-MM-DD'), 1, 1, 'Aliyar Aghayev', 40, 2, 2, 5);


-- Trigger for Teams table to enforce a constraint on the manager's name length
CREATE OR REPLACE TRIGGER check_manager_name_length
BEFORE INSERT OR UPDATE ON teams
FOR EACH ROW
```

```
78 BEGIN
79     IF LENGTH(:new.manager_name) > 80 THEN
80         RAISE_APPLICATION_ERROR(-20001, 'Manager name cannot exceed 80
       characters.');
81     END IF;
82 END;
83
84 -- Example to check the trigger
85 -- INSERT INTO teams VALUES (2132,'Barcelona',TO_DATE('1899-11-29', 'YYYY-
       MM-DD'),1,'Xavierddddddddddsdfsdfsddddddddddddddddddddddddddddddddddddddd
86 -- ddddddddddddddddddddddddddddddd Hernandez Creus');
```

Listing 6: Triggers (`DBTriggers.sql` file)

2.7 Package

```
1 -- Package containing procedure for updating the capacity of specific
       stadium and procedure for generation of report with match scores
2 CREATE OR REPLACE PACKAGE football_package IS
3     PROCEDURE update_stadium_capacity (
4         stadium_id   IN NUMBER,
5         new_capacity IN NUMBER
6     );
7
8     PROCEDURE generate_match_scores_report (
9         match_week IN NUMBER
10    );
11
12 END football_package;
13 /
14
```

```
15 CREATE OR REPLACE PACKAGE BODY football_package IS
16
17     PROCEDURE update_stadium_capacity (
18         stadium_id   IN NUMBER,
19         new_capacity IN NUMBER
20     ) IS
21     BEGIN
22         UPDATE stadiums
23         SET capacity = new_capacity
24         WHERE id = stadium_id;
25
26     END update_stadium_capacity;
27
28
29     PROCEDURE generate_match_scores_report (
30         match_week IN NUMBER
31     ) IS
32         CURSOR match_scores_cursor IS
33         SELECT m.id,
34                m.match_date,
35                m.home_team_score,
36                m.away_team_score,
37                ht.name AS home_team_name,
38                at.name AS away_team_name
39         FROM matches m
40             INNER JOIN teams ht ON m.home_team_id = ht.id
41             INNER JOIN teams at ON m.away_team_id = at.id
42         WHERE m.match_week = match_week;
43
44     BEGIN
45         FOR match_score IN match_scores_cursor LOOP
46             dbms_output.put_line('Match ID: ' || match_score.id);
47             dbms_output.put_line('Match Date: ' || match_score.match_date);
48             dbms_output.put_line('Home Team: ' || match_score.
```

```
    home_team_name);
49              dbms_output.put_line('Home Team Score: ' || match_score.
    home_team_score);
50              dbms_output.put_line('Away Team: ' || match_score.
    away_team_name);
51              dbms_output.put_line('Away Team Score: ' || match_score.
    away_team_score);
52              dbms_output.put_line('------------------------');
53          END LOOP;
54      END generate_match_scores_report;
55
56  END football_package;
57
58  -- Example of update_stadium_capacity usage from the football_package
59  -- BEGIN
60  --     football_package.update_stadium_capacity(1, 20000);
61  -- END;
62
63  -- Example of generate_match_scores_report usage from the football_package
64  -- BEGIN
65  --     football_package.generate_match_scores_report(12);
66  -- END;
67
```

```
------------------------
Match ID: 103
Match Date: 24/04/27
Home Team: Osasuna
Home Team Score: 1
Away Team: Girona
Away Team Score: 1
------------------------
```

Figure 22: Procedure `generate_match_scores_report` output

Listing 7: Package (`DBPackage.sql` file)

2.8 Other additional database structures

```sql
-- Simple view to show members' stats
CREATE VIEW member_stats_view AS
    SELECT m.id,
           m.name,
           m.surname,
           m.dob,
           m.salary,
           m.nationality,
           t.name                              AS team_name,
           COUNT(DISTINCT ma.id)               AS matches_played,
           COUNT(DISTINCT ma.id) * m.salary AS total_earnings,
           SUM(CASE
                   WHEN ma.home_team_id = m.team_id THEN
                       ma.home_team_score
                   ELSE
                       ma.away_team_score
               END) AS goals_scored
    FROM members m
        JOIN teams t ON m.team_id = t.id
        LEFT JOIN matches ma ON m.team_id = ma.home_team_id
                                OR m.team_id = ma.away_team_id
    GROUP BY m.id,
             m.name,
             m.surname,
             m.dob,
             m.salary,
             m.nationality,
             t.name;

-- Example of member_stats_view usage
-- SELECT * FROM member_stats_view;
```

ID	NAME	SURNAME	DOB	SALARY	NATIONALITY	TEAM_NAME	MATCHES_PLAYED	TOTAL_EARNINGS	GOALS_SCORED
1	Marc-Andre	ter Stegen	92/04/30	9791,089	German	Barcelona	52	509136,628	56
2	Ronald	Araujo	99/03/07	8643,899	Uruguayan	Barcelona	52	449482,748	56
3	Jules	Kounde	98/11/12	7596,578	French	Barcelona	52	395022,056	56
4	Andreas	Christensen	96/04/10	8238,23	Danish	Barcelona	52	428387,96	56

Figure 23: View `member_stats_view` output

```
32
33
34 -- XML report
35 DECLARE
36     xml_data XMLTYPE;
37 BEGIN
38     xml_data := xmltype('<stadiums>
39                          <stadium>
40                            <id>11</id>
41                            <name>ttt</name>
42                            <city>test</city>
43                            <capacity>12374</capacity>
44                          </stadium>
45                          <stadium>
46                            <id>12</id>
47                            <name>Sds</name>
48                            <city>Ssdf</city>
49                            <capacity>32774</capacity>
50                          </stadium>
51                        </stadiums>');
52
53     FOR s IN (
54         SELECT
55             extractvalue(value(st),
56                          '/stadium/id')        AS id,
57             extractvalue(value(st),
58                          '/stadium/name')      AS name,
59             extractvalue(value(st),
```

```
                          '/stadium/city')     AS city,
            extractvalue(value(st),
                          '/stadium/capacity') AS capacity
        FROM
          TABLE ( xmlsequence(xml_data.extract('/stadiums/stadium')) ) st
    ) LOOP
        INSERT INTO stadiums (id,
                              name,
                              city,
                              capacity
        ) VALUES (s.id,
                  s.name,
                  s.city,
                  s.capacity);
    END LOOP;

    COMMIT;
END;

-- SELECT * FROM stadiums;


-- Pipelined function
CREATE OR REPLACE TYPE ticket_profit_type AS OBJECT (
    match_id NUMBER,
    profit   NUMBER
);
/

CREATE OR REPLACE TYPE ticket_profit_table_type AS
    TABLE OF ticket_profit_type;
/

```

```
93  CREATE OR REPLACE FUNCTION calculate_ticket_profit RETURN
        ticket_profit_table_type
94      PIPELINED
95  AS
96      v_profit NUMBER;
97  BEGIN
98      FOR rec IN (SELECT t.match_id,
99                         ( t.price * t.quantity ) AS profit
100                FROM tickets t
101     ) LOOP
102         v_profit := rec.profit;
103         PIPE ROW ( ticket_profit_type(rec.match_id, v_profit) );
104     END LOOP;
105
106     RETURN;
107 END;
108
109 SELECT match_id,
110        profit
111 FROM
112   TABLE ( calculate_ticket_profit );
113
```

MATCH_ID	PROFIT
89	1453500

Figure 24: Pipelined function `calculate_ticket_profit` output

Listing 8: Other additional structures (`DBAdditionalStructures.sql` file)