AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Assignment Title:	Sports Club Management System		
Assignment No:		Date of Submission:	27/12/2022
Course Title:	Advance Database Management System		
Course Code:	CSC4181	Section:	B
Semester:	Fall 2022-23	Course Teacher:	JUENA AHMED NOSHIN

Declaration and Statement of Authorship:

1. I/we hold a copy of this Assignment/Case-Study, which can be produced if the original is lost/damaged.
2. This Assignment/Case-Study is my/our original work and no part of it has been copied from any other student's work or from any other source except where due acknowledgement is made.
3. No part of this Assignment/Case-Study has been written for me/us by any other person except where such collaborations been authorized by the concerned teacher and is clearly acknowledged in the assignment.
4. I/we have not previously submitted or currently submitting this work for any other course/unit.
5. This work may be reproduced, communicated, compared and archived for the purpose of detecting plagiarism.
6. I/we give permission for a copy of my/our marked work to be retained by the Faculty for review and comparison, including review by external examiners.
7. I/we understand thatPlagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a form of cheating and is a very serious academic offense that may lead to expulsion from the University. Plagiarized material can be drawn from, and presented in, written, graphic and visual form, including electronic data, and oral presentations. Plagiarism occurs when the origin of them arterial used is not appropriately cited.
8. I/we also understand that enabling plagiarism is the act of assisting or allowing another person to plagiarize or to copy my/our work.

* *Student(s) must complete all details except the faculty use part.*
** Please submit all assignments to your course teacher or the office of the concerned teacher.

Group Name/No.:

No	Name	ID	Program
1	Yeasin Elahi	19-39872-1	BSc CSE
2	Khaled Mansur	19-39861-1	BSc CSE
3	MD. Mahbub Alam Siddik	19-39376-1	BSc CSE
4	Arif Hossen	19-40741-1	BSc CSE

Faculty use only		
FACULTYCOMMENTS	**Marks Obtained**	
	Total Marks	

CONTENTS

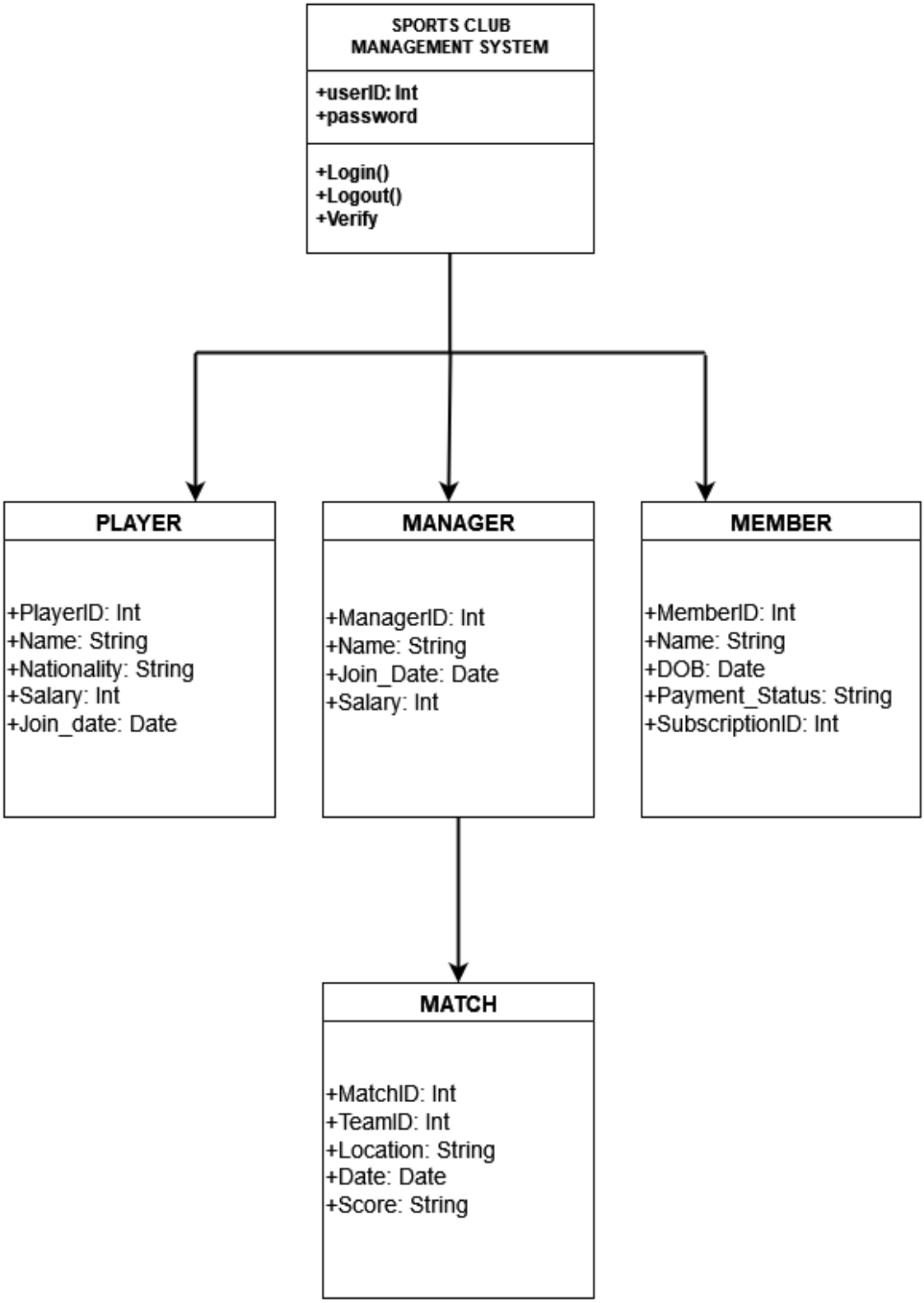
INTRODUCTION

Today we are living in a world of technology. We are using technology as our daily needs. Now there are many sports clubs that formed for different kind of activities. And there are also many variations of clubs that running across the world. In this management system two role Admin and Members. Admin can arrange all the necessary things for the club. Admin can create the players, manager and staff's profile. And the members need to create their own profile by registration. Members can edit their profile by own. They can see the match schedule, players stats and manager details. They also can buy the match ticket from the site. In players profile we can see their stats like match played, goals, assists etc.

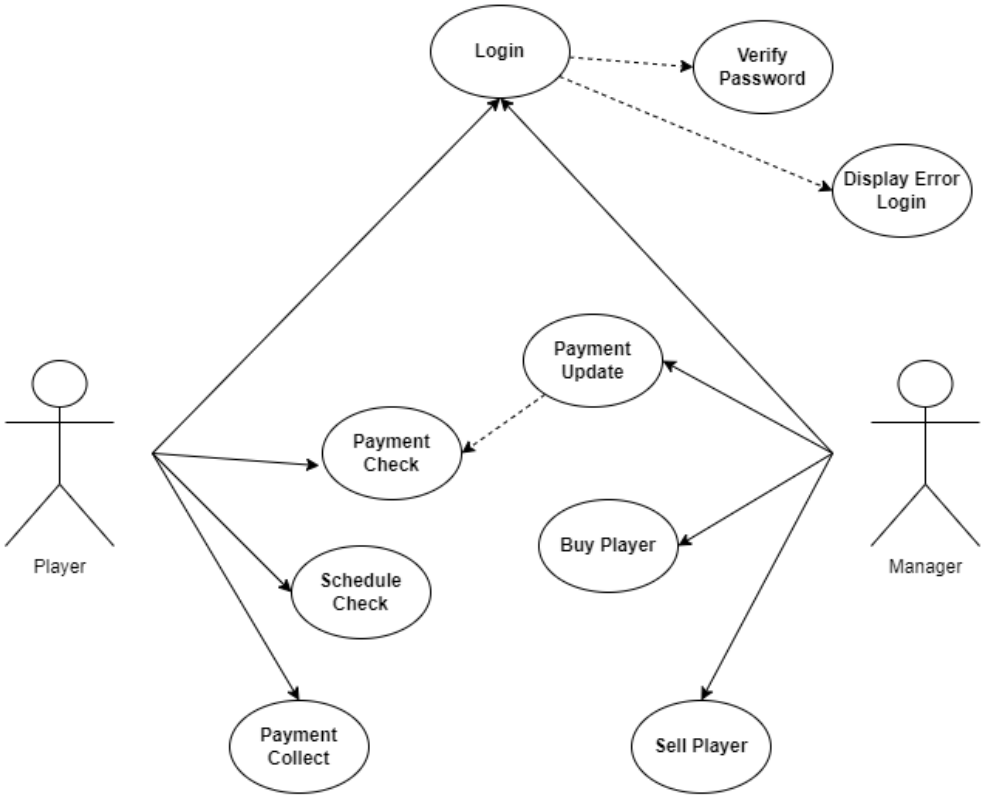
PROJECT PROPOSAL

In this project we build a sports club management system project. And we working for football club management system. In football club management systems, there are 2 types of users. First one is Admin. Admin can control the whole systems. Admin can create profile for the players, manager and also for staffs. Admin also delete profiles who leaves the club. Admin can post the notice on the website. Admin also control the ticket processing. Admin always update the sites. Admin always update players stats. Admin also post notice about the players injury update and post the playing XI and whole squad players name before the match. And the other users are Members who are supported the club. Members need to create their profile by own. They can register by their name, number, email, age, address. They can see the players and manager details on the site. Members also see the staffs who working for the club. They can see the match schedule and buy tickets for the match from the site. Members also see players stats like match played, goals number, assist number, injury update etc.

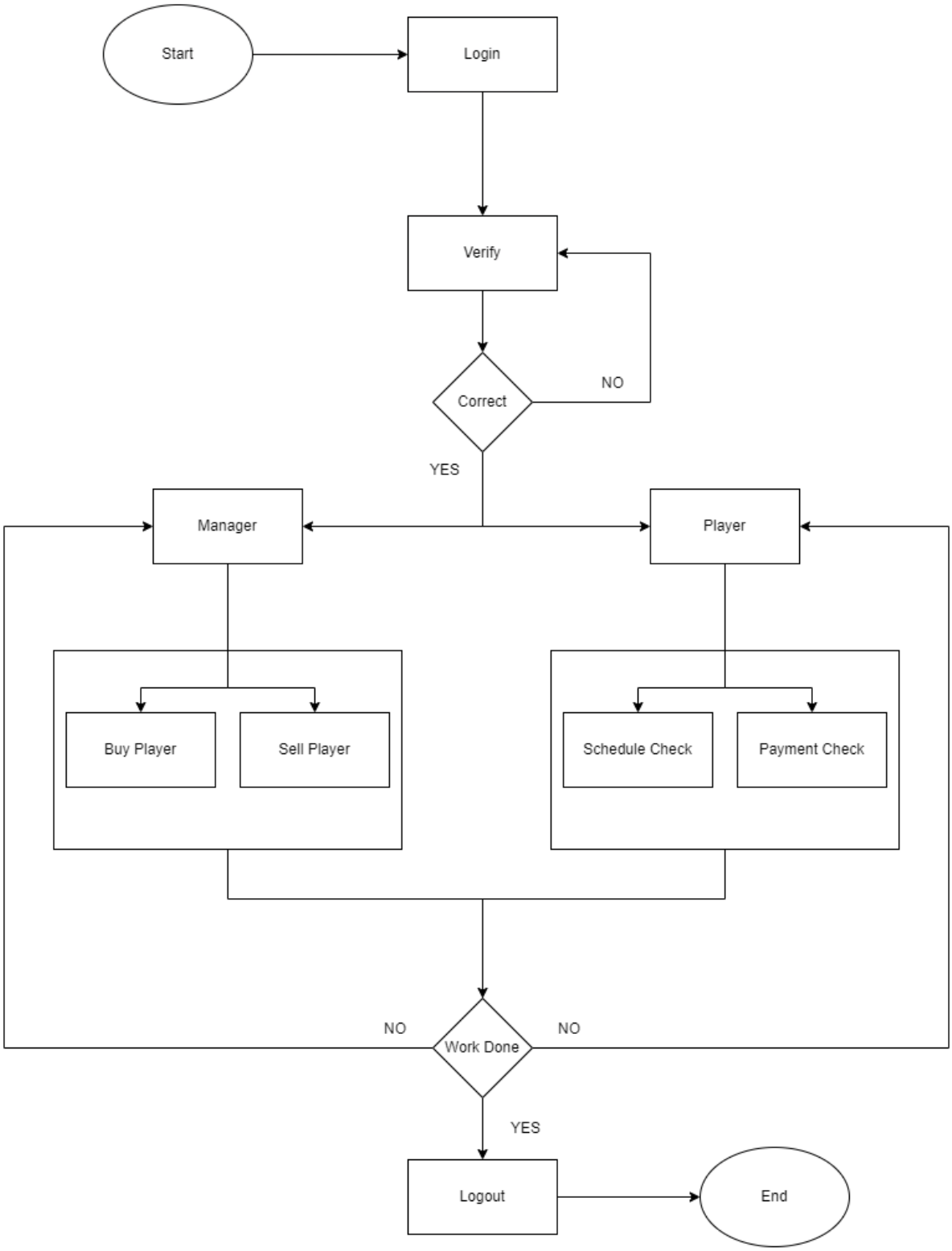
CLASS DIAGRAM



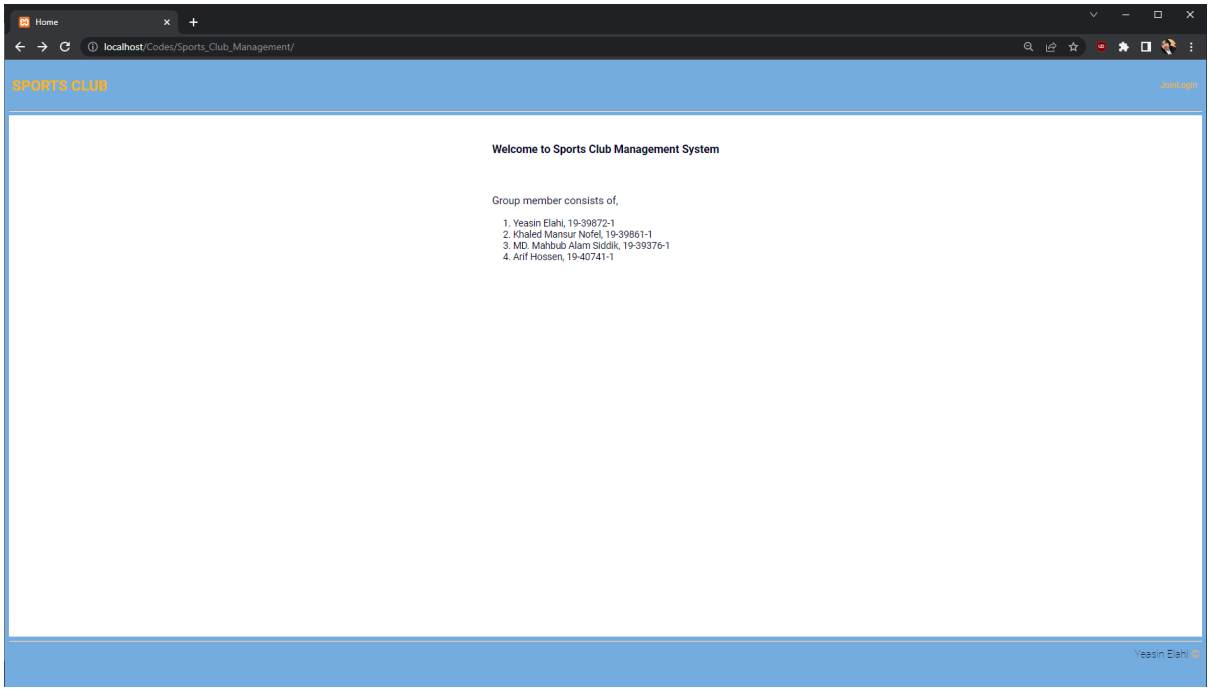
USE CASE DIAGRAM

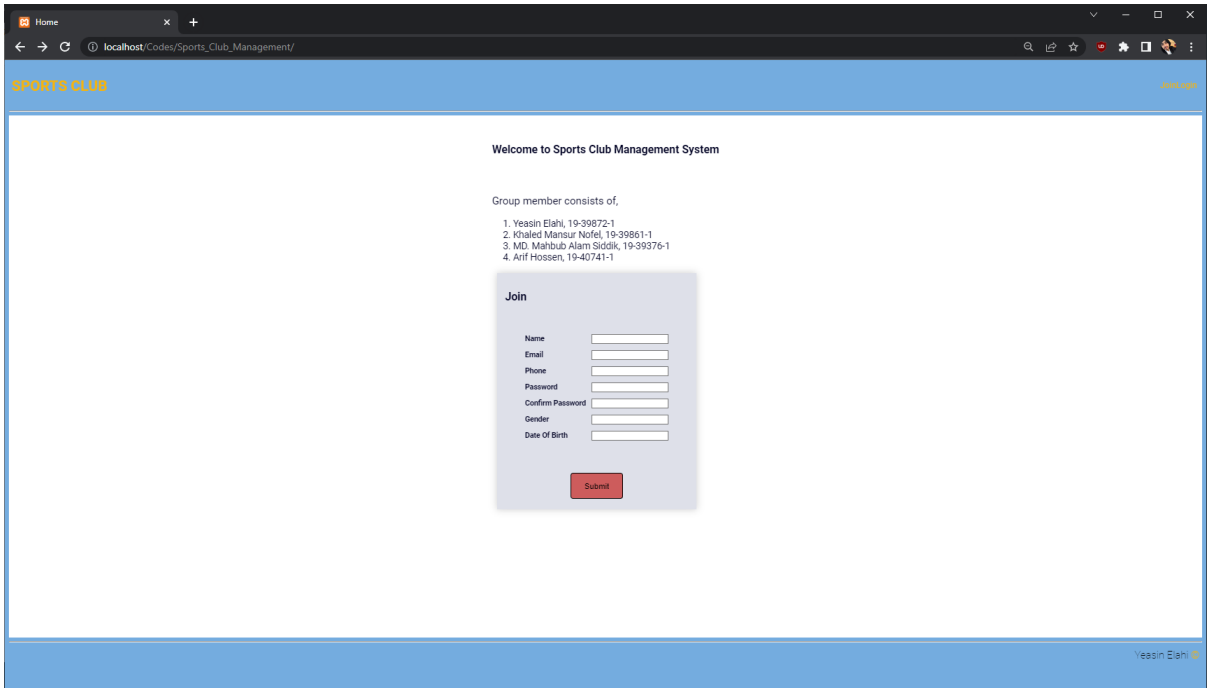


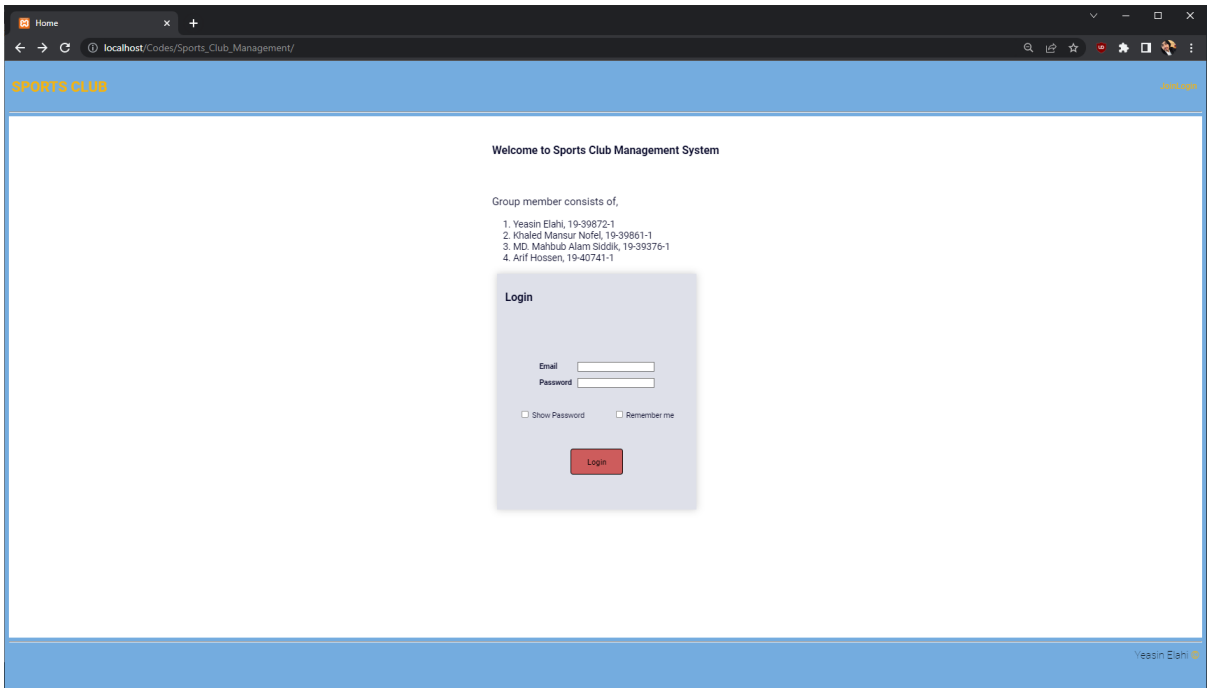
ACTIVITY DIAGRAM

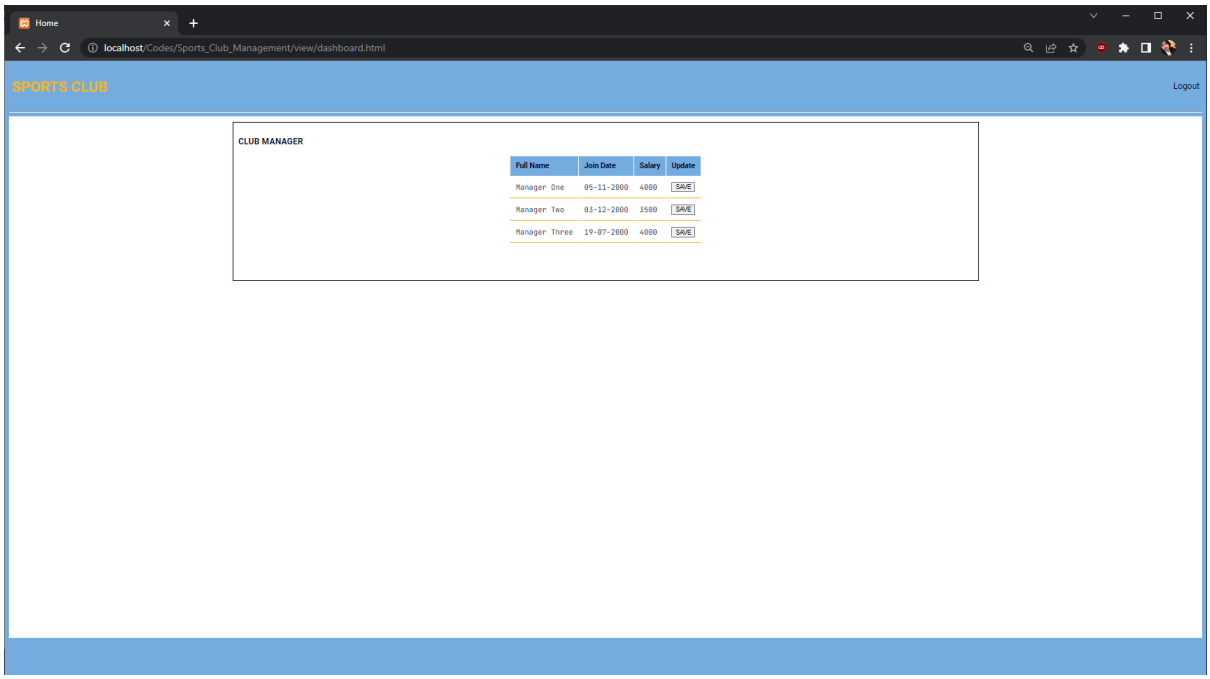


USER INTERFACE






Home
localhost/Codes/Sports_Club_Management/
SPORTS CLUB
Welcome to Sports Club Management System
Group member consists of,
1. Yeasin Elahi, 19-39872-1
2. Khaled Mansur Nofel, 19-39861-1
3. MD. Mahbub Alam Siddik, 19-39376-1
4. Arif Hossen, 19-40741-1
Login
Email
Password
Show Password
Remember me
Login
Yeasin Elahi


Home
localhost/Codes/Sports_Club_Management/view/dashboard.html
SPORTS CLUB
Logout
CLUB MANAGER
Full Name
Join Date
Salary
Update
Manager One
05-11-2000
4000
SAVE
Manager Two
03-12-2000
3500
SAVE
Manager Three
19-07-2000
4000
SAVE

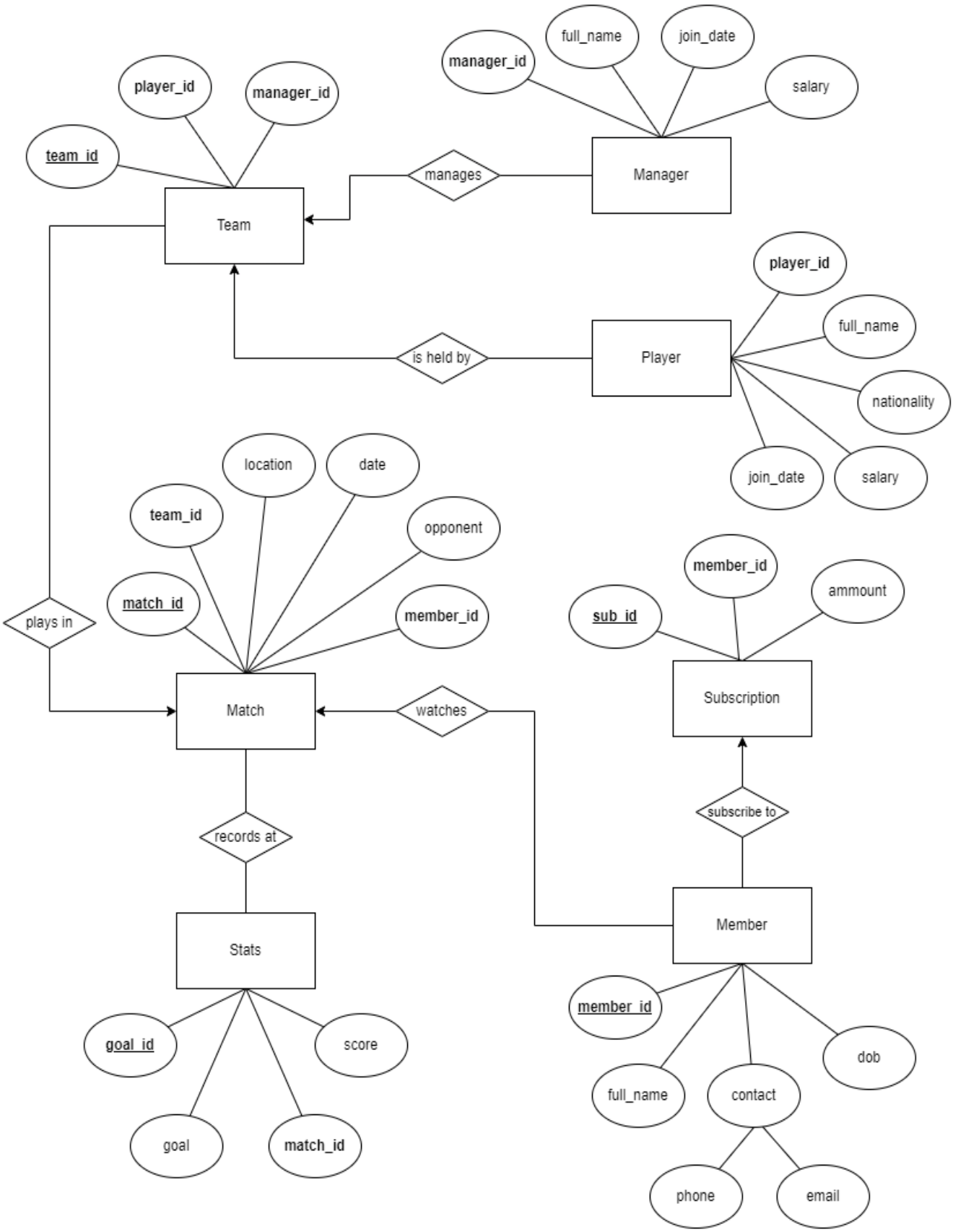
SCENARIO DESCRIPTION

A sports club is maintained by club owner, where he is identified by **(id, name, password)**. Club owner can assign manager to the club where manager is identified by **(manager_id, name, join_date, salary)**. Further more, a club owner can buy players for his club. A player can be identified by **(player_id, name, nationality, join_date, salary)**. Manager manages a team and players are held by a team. A team can be identified by **(team_id, player_id, manager_id)**.

A member can join the club with a membership. A member can be identified by **(member_id, name, phone, email, dob)**. Members can watch matches with discount if they had a subscription. A member can subscribe to a subscription where it is identified by **(sub_id, member_id amount)**.

A team plays at a match and members can watch matches. A match is identified by **(match_id, team_id, location, date, opponent, member_id)**. A match is recoded at stats where stats can be identified by **(goal_id, goal, match_id, score)**.

ER DIAGRAM


manager_id
full_name
join_date
salary
player_id
manager_id
team_id
manages
Manager
Team
player_id
full_name
is held by
Player
nationality
join_date
salary
location
date
team_id
opponent
match_id
member_id
plays in
member_id
ammount
sub_id
Match
watches
Subscription
subscribe to
records at
Member
Stats
member_id
goal_id
score
dob
full_name
contact
goal
match_id
phone
email

NORMALIZATION

Manages~

UNF

Manages (manager_id, full_name, join_date, salary, team_id, player_id, manager_id)

1NF

- manager_id, full_name, join_date, salary, team_id, player_id, manager_id

2NF

- manager_id, full_name, join_date, salary
- team_id, player_id, manager_id

3NF

There is no transitive dependency, Relation already in 3NF.

- manager_id, full_name, join_date, salary
- team_id, player_id, manager_id

Table

1. manager_id, full_name, join_date, salary
2. team_id, player_id, manager_id

Is Held By~

UNF

Is Held By (player_id, full_name, nationality, salary, join_date, team_id, player_id, manager_id)

1NF

- player_id, full_name, nationality, salary, join_date, team_id, player_id, manager_id

2NF

- player_id, full_name, nationality, salary, join_date
- team_id, player_id, manager_id

3NF

There is no transitive dependency, Relation already in 3NF.

- player_id, full_name, nationality, salary, join_date
- team_id, player_id, manager_id

Table

1. player_id, full_name, nationality, salary, join_date
2. team_id, player_id, manager_id

Subscribe To~

UNF

Subscribe To (sub_id, member_id, amount, member_id, full_name, dob, phone, email)

1NF

Phone Number is a multivalued attribute.

- sub_id, member_id, amount, member_id, full_name, dob, phone, email

2NF

- sub_id, member_id, amount
- member_id, full_name, dob, phone, email

3NF

There is no transitive dependency, Relation already in 3NF.

- sub_id, member_id, amount
- member_id, full_name, dob, phone, email

Table

1. sub_id, member_id, amount
2. member_id, full_name, dob, phone, email

Plays At~

UNF

Plays At (team_id, player_id, manager_id, match_id, team_id, location, date, opponent, member_id)

1NF

- team_id, player_id, manager_id, match_id, team_id, location, date, opponent, member_id

2NF

- team_id, player_id, manager_id
- match_id, team_id, location, date, opponent, member_id

3NF

There is no transitive dependency, Relation already in 3NF.

- team_id, player_id, manager_id
- match_id, team_id, location, date, opponent, member_id

Table

1. team_id, player_id, manager_id
2. match_id, team_id, location, date, opponent, member_id

Watches~

UNF

Watches (match_id, team_id, location, date, opponent, member_id, member_id, full_name, dob, phone, email)

1NF

Phone Number is a multivalued attribute.

- match_id, team_id, location, date, opponent, member_id, member_id, full_name, dob, phone, email

2NF

- match_id, team_id, location, date, opponent, member_id
- member_id, full_name, dob, phone, email

3NF

There is no transitive dependency, Relation already in 3NF.

- match_id, team_id, location, date, opponent, member_id
- member_id, full_name, dob, phone, email

Table

1. match_id, team_id, location, date, opponent, member_id
2. member_id, full_name, dob, phone, email

Records At~

UNF

Records At (match_id, team_id, location, date, opponent, member_id, goal_id, goal, match_id, score)

1NF

- match_id, team_id, location, date, opponent, member_id, goal_id, goal, match_id, score

2NF

- match_id, team_id, location, date, opponent, member_id
- goal_id, goal, match_id, score

3NF

There is no transitive dependency, Relation already in 3NF.

- match_id, team_id, location, date, opponent, member_id
- goal_id, goal, match_id, score

Table

1. match_id, team_id, location, date, opponent, member_id
2. goal_id, goal, match_id, score

SCHEMA DIAGRAM

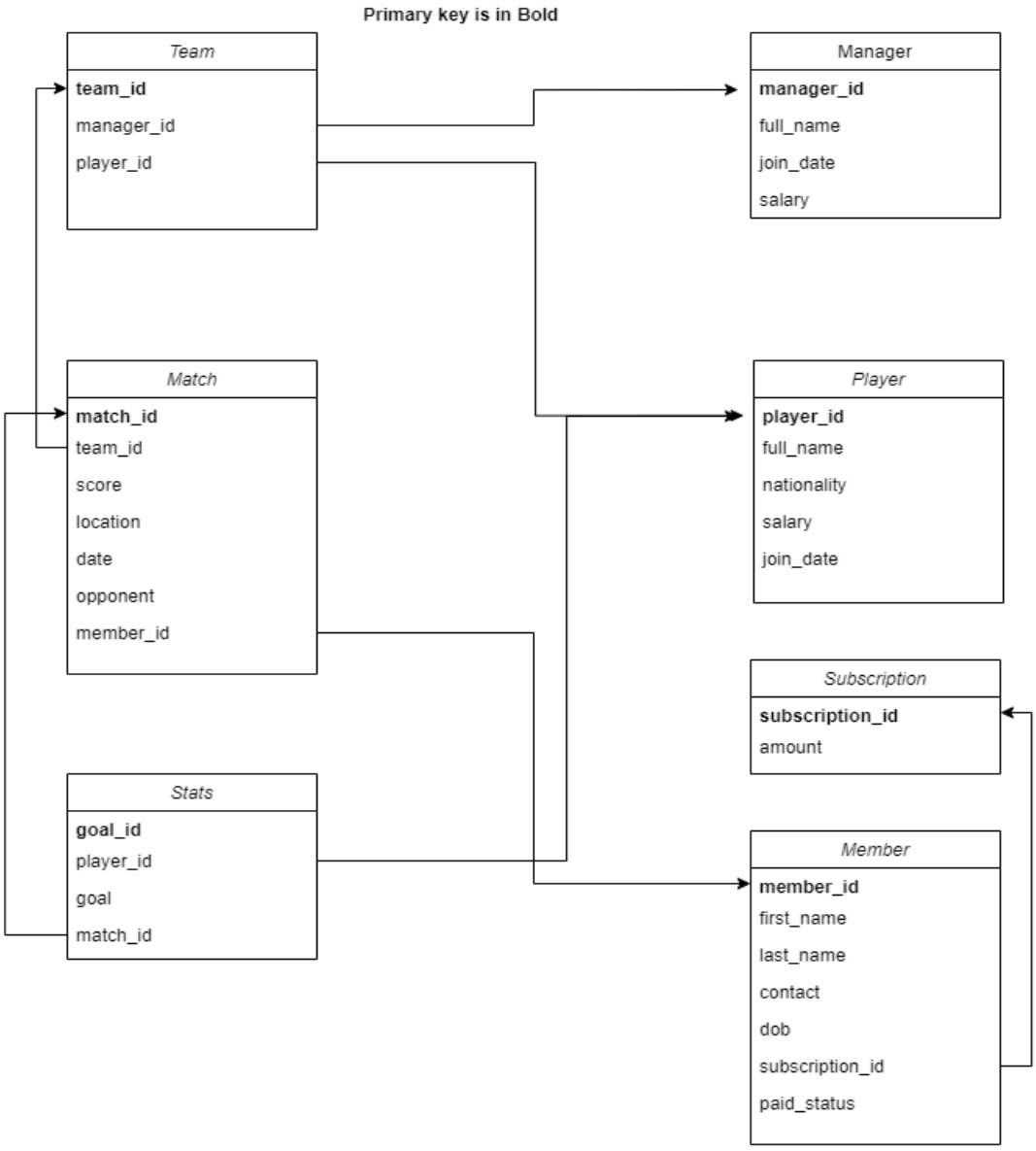


TABLE CREATION

Player

create table player(player_id number(5) not null, full_name varchar(50) not null, nationality varchar(15), salary number(15), join_date date, primary key(player_id));

Results Explain Describe Saved SQL History

Object Type TABLE Object PLAYER

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PLAYER	PLAYER_ID	Number	-	5	0	1	-	-	-
	FULL_NAME	Varchar2	50	-	-	-	-	-	-
	NATIONALITY	Varchar2	15	-	-	-	✓	-	-
	SALARY	Number	-	15	0	-	✓	-	-
	JOIN_DATE	Date	7	-	-	-	✓	-	-

1 - 5

Manager

create table manager(manager_id number(5) not null, full_name varchar(50) not null, join_date date, salary number(15), primary key(manager_id));

Results Explain Describe Saved SQL History

Object Type TABLE Object MANAGER

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MANAGER	MANAGER_ID	Number	-	5	0	1	-	-	-
	FULL_NAME	Varchar2	50	-	-	-	-	-	-
	JOIN_DATE	Date	7	-	-	-	✓	-	-
	SALARY	Number	-	15	0	-	✓	-	-

1 - 4

Team

create table team(team_id number(5) not null, player_id number(5) not null, manager_id number(5) not null, primary key(team_id), foreign key(player_id) references player(player_id), foreign key(manager_id) references manager(manager_id));

Results Explain Describe Saved SQL History

Object Type TABLE Object TEAM

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
TEAM	TEAM_ID	Number	-	5	0	1	-	-	-
	PLAYER_ID	Number	-	5	0	-	-	-	-
	MANAGER_ID	Number	-	5	0	-	-	-	-

1 - 3

Member

create table member(member_id number(10) not null, full_name varchar(50) not null, phone number(10), email varchar(30) not null, dob date, primary key(member_id));

Results Explain Describe Saved SQL History

Object Type TABLE Object MEMBER

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MEMBER	MEMBER_ID	Number	-	5	0	1	-	-	-
	FULL_NAME	Varchar2	50	-	-	-	-	-	-
	PHONE	Number	-	10	0	-	✓	-	-
	EMAIL	Varchar2	30	-	-	-	-	-	-
	DOB	Date	7	-	-	-	✓	-	-

1 - 5

Subscription

create table subscription(subscription_id number(5) not null, member_id number(5) not null,amount number(15), primary key(subscription_id), foreign key(member_id) references member(member_id));

Results Explain Describe Saved SQL History

Object Type TABLE Object SUBSCRIPTION

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
SUBSCRIPTION	SUBSCRIPTION_ID	Number	-	5	0	1	-	-	-
	MEMBER_ID	Number	-	5	0	-	-	-	-
	AMOUNT	Number	-	15	0	-	✓	-	-

1 - 3

Match

create table match(match_id number(5) not null, team_id number(5) not null, location varchar(20) not null, match_date date not null, opponent varchar(20) not null, member_id number(5) not null, primary key(match_id), foreign key(team_id) references team(team_id), foreign key(member_id) references member(member_id));

Results Explain Describe Saved SQL History

Object Type TABLE Object MATCH

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MATCH	MATCH_ID	Number	-	5	0	1	-	-	-
	TEAM_ID	Number	-	5	0	-	-	-	-
	LOCATION	Varchar2	20	-	-	-	-	-	-
	MATCH_DATE	Date	7	-	-	-	-	-	-
	OPPONENT	Varchar2	20	-	-	-	-	-	-
	MEMBER_ID	Number	-	5	0	-	-	-	-

1 - 6

Stats

create table stats(goal_id number(5) not null, goal number(2) not null, match_id number(5) not null, score varchar(5) not null, primary key(goal_id), foreign key(match_id) references match(match_id));

Results Explain Describe Saved SQL History

Object Type TABLE Object STATS

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
STATS	GOAL_ID	Number	-	5	0	1	-	-	-
	GOAL	Number	-	2	0	-	-	-	-
	MATCH_ID	Number	-	5	0	-	-	-	-
	SCORE	Varchar2	5	-	-	-	-	-	-

1 - 4

DATA INSERTION

Player

insert into player(player_id, full_name, nationality, salary, join_date) values(1001, 'Ricardo Kaka', 'Brazillian', 1500, '12-apr-2021');

insert into player(player_id, full_name, nationality, salary, join_date) values(1002, 'Jamal Bhuiya', 'Bangladeshi', 580, '10-jul-2021');

insert into player(player_id, full_name, nationality, salary, join_date) values(1003, 'Tapu Barman', 'Bangladeshi', 500, '07-aug-2021');

insert into player(player_id, full_name, nationality, salary, join_date) values(1004, 'Eder Militao', 'Brazillian', 1300, '07-sep-2021');

insert into player(player_id, full_name, nationality, salary, join_date) values(1005, 'Neymar Jr', 'Brazillian', 1800, '07-oct-2021');

Results Explain Describe Saved SQL History

PLAYER_ID	FULL_NAME	NATIONALITY	SALARY	JOIN_DATE
1001	Ricardo Kaka	Brazillian	1500	12-APR-21
1002	Jamal Bhuiya	Bangladeshi	580	10-JUL-21
1003	Tapu Barman	Bangladeshi	500	07-AUG-21
1004	Eder Militao	Brazillian	1300	07-SEP-21
1005	Neymar Jr	Brazillian	1800	07-OCT-21

5 rows returned in 0.00 seconds CSV Export

Manager

insert into manager(manager_id, full_name, join_date, salary) values(101, 'manager one', '1-oct-2000', 5000);

insert into manager(manager_id, full_name, join_date, salary) values(102, 'Jose Mourinho', '10-apr-2020', 1500);

insert into manager(manager_id, full_name, join_date, salary) values(103, 'Carlo Ancelotti', '10-may-2021', 1500);

Results Explain Describe Saved SQL History

MANAGER_ID	FULL_NAME	JOIN_DATE	SALARY
101	manager one	01-JAN-00	5000
102	Jose Mourinho	10-APR-20	1500
103	Carlo Ancelotti	10-MAY-21	1500

3 rows returned in 0.00 seconds CSV Export

Team

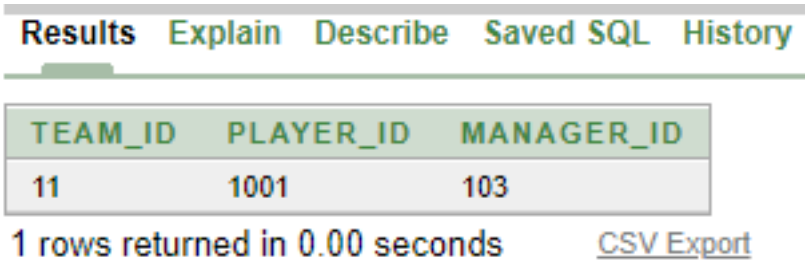
insert into team(team_id, player_id, manager_id) values(11, 1001, 103);Member

insert into member(member_id, full_name, phone, email, dob) values(10001, 'Arif Hossen', 1874975923, 'arif@gmail.com', '25-nov-2000');

insert into member(member_id, full_name, phone, email, dob) values(10002, 'Rimo Khan', 1365890754, 'rimo@gmail.com', '22-nov-2004');

insert into member(member_id, full_name, phone, email, dob) values(10003, 'Asif Khan', 1487296510, 'asif@gmail.com', '19-oct-1998');

insert into member(member_id, full_name, phone, email, dob) values(10004, 'Alif Ali', 9188761260, 'alif@gmail.com', '9-aug-2001');



Subscription

insert into subscription(subscription_id, member_id, amount) values(51, 10004, 25);

insert into subscription(subscription_id, member_id, amount) values(52, 10002, 50);

insert into subscription(subscription_id, member_id, amount) values(53, 10003, 75);

Results Explain Describe Saved SQL History

SUBSCRIPTION_ID	MEMBER_ID	AMOUNT
51	10004	25
52	10002	50
53	10003	75

3 rows returned in 0.00 seconds CSV Export

Match

insert into match(match_id, team_id, location, match_date, opponent, member_id) values(1, 11, 'Real Madrid', '27-dec-2022', 'Real Madrid CF',10002);

Results Explain Describe Saved SQL History

MATCH_ID	TEAM_ID	LOCATION	MATCH_DATE	OPPONENT	MEMBER_ID
1	11	Real Madrid	27-DEC-22	Real Madrid CF	10002

1 rows returned in 0.00 seconds CSV Export

Stats

insert into stats(goal_id, goal, match_id, score) values(1, 6, 1, 'lose');

Results Explain Describe Saved SQL History

GOAL_ID	GOAL	MATCH_ID	SCORE
1	6	1	lose

1 rows returned in 0.00 seconds CSV Export

QUERY WRITING

Single row function

Q1: Use TO_CHAR function with dates to show name and joining date

Answer

```
SELECT full_name,
TO_CHAR(join_date, 'fmDD Month YYYY') join_date
FROM Manager;
```

Q2: Use CONCAT function to concatenate two string values

Answer:

```
SELECT CONCAT (full_name,salary)
FROM Manager WHERE rownum <6;
```

Q3: Calculate how many days have passed since joining using the date arithmetic function

Answer

```
SELECT manager_id, (sysdate - join_date) Passed_days
FROM Manager WHERE rownum < 5;
```

Group Function

Q1: Display the player name which is more costly than 100000 and group by its name

Answer

```
SELECT full_name

FROM Player

GROUP BY full_name

HAVING salary>1000;
```

Q2: Display the Item_name which is more costlier than egg and group by its name

Answer

```
SELECT full_name

FROM Player

GROUP BY full_name

HAVING salary<100000;
```

Q3: Display the oldest and latest joining date of player

Answer

```
SELECT MIN (join_date) oldest, MAX (join_date) latest

FROM Player;
```

Sub Query

Q1: Display the palyer name who take "120000" salary

Answer

Select * from Player;

select full_name from Player where salary = (select salary from Player where salary='120000');

Q2: Display which player join earlier than Neymar

Answer

SELECT full_name,player_id

FROM Player

WHERE join_date< (SELECT join_date

FROM player WHERE palyer_id=100);

Q3: Display the palyer name who take "120000" salary

Answer:

Select * from Player;

select full_name from Player where salary < (select salary from Player where salary<'1200');

Joining

Q1: Display the name of the player who played match 11

Answer

```
SELECT Player.full_Name

from player ,match_id

where Player.player_id=match_id.player_id and match_id.player_id='11';
```

Q2: Display the name of the player which one belongs to captain "true"

Answer

```
SELECT Player.full_name

from Player,captain

where player.player_id=player.player_id and team.captain='true';
```

Q3: Display the name of the player who score "2" in the particular match

Answer

```
SELECT Player.full_name from Player,Stats where

Player.player_id=Match.match_id and Stats.match_id =3;
```

View

Q1: Create a view called StatsView based on the match_id from the Stats table.

Answer

```
CREATE VIEW StatsView

AS SELECT match_id FROM Stats

WHERE match_id = '11';

Select * from StatsView;
```

Q2: Create a view called PlayerView based on the full_name and salary from the Player table.

Answer

```
CREATE VIEW PlayerView

AS SELECT full_name,salary FROM Player WHERE Id = 1;

Select * from PlayerView;
```

Q3: Create a view called ManagerInfo that contains full_name and salary

Answer

```
CREATE VIEW ManagerInfo AS SELECT full_name , salary

FROM Manager where Id=1;
```

Synonym

Q1: Create a synonym for view called PlayerInfo

Answer

CREATE SYNONYM Pinfo

For PlayerInfo;

Q2: Create a synonym for view called PlayerView

Answer

CREATE SYNONYM PView

For PlayerView;

Q3: Create a synonym for view called StatsView

Answer

CREATE SYNONYM SView

For StatsView;

Function

Q1: Defining and Invoking a simple PL/SQL function which will compute and return the total number of Player quantity.

Ans:

```
CREATE OR REPLACE FUNCTION TotalPlayer
RETURN number IS
total number(26) := 0;
BEGIN
SELECT count(*) into total
FROM Player;
RETURN total;
END;
/
DECLARE
c number(26);
BEGIN
c := TotalPlayer();
dbms_output.put_line('Total Player quantity: ' || c);
END;
```

Q2: Create a function that takes the name as input and returns the welcome message as output. We are going to use anonymous block and select statement to call the function.

Ans:

```
CREATE OR REPLACE FUNCTION welcome_message ( p_name IN VARCHAR2)
RETURN VARCHAR2

IS

BEGIN

RETURN ('Welcome '|| p_name);

END;

/

DECLARE

lv_message VARCHAR2(250);

BEGIN

lv_message := welcome_message ('Militao');

dbms_output.put_line(lv_message);

END;
```

Q3: Q3: Create a PL/SQL function get_player_id that returns the player’s Id given Player name and also show exception message when data did not found.

Ans:

```
REATE OR REPLACE FUNCTION get_player_id(player_id IN VARCHAR)

RETURN NUMBER

IS

pid VARCHAR2 (50);

BEGIN

SELECT Player_ID INTO pid FROM Player WHERE P_NAME= 'P8';
```

```
RETURN cid;

EXCEPTION

WHEN no_data_found THEN

DBMS_OUTPUT.PUT_LINE('NO SUCH Player');

RETURN pid;

END;

/

DECLARE

pid VARCHAR2 (50);

BEGIN

pid := get_player_id (1010);

DBMS_OUTPUT.PUT_LINE (pid);

END;
```

Procedure

Q1: Adjust value of player using procedure.

Ans:

```
CREATE OR REPLACE PROCEDURE adjust_value(

in_Value IN Player.Value%TYPE

)

IS

BEGIN

UPDATE Player

SET Value='$50'

WHERE Value= in_Value;

END;

begin

adjust_Value('$50');

end

select * from Value;

rollback
```

Q2: Increase the quantity of selling Jersey of transaction table using procedure.

Ans:

```
CREATE OR REPLACE PROCEDURE adjust_Quantity(

in_Quantity IN Transaction.Quantity%TYPE

)
```

```
IS

BEGIN

UPDATE Transaction

SET Quantity='5'

WHERE Quantity= in_Quantity ;

END;

begin

adjust_Quantity('3');

end

select * from Transaction;

rollback
```

Q3: Display your project title(…) using procedure

Ans:

```
CREATE OR REPLACE PROCEDURE project_title

AS

BEGIN

dbms_output.put_line('Sports Club Management System');

END;

/

BEGIN

project_title;

END;

/
```

Record

Q1: Display a player id using record (one row)

Ans:

```
declare
Player_rec Player%rowtype;
begin
select * into Player_rec from Player
where p_name='p1';
dbms_output.put_line(Player_rec.p_id);
end
/
```

Q2: Display manager name and their addresses using record (multiple row)

Ans:

```
declare
Manager_rec Manager%rowtype;
begin
for Manager_rec
in(select * from Manager)
loop
dbms_output.put_line(Manager_rec.Manager_Name||' '||Manager_rec.e_Address);
end loop;
end
```

/

Q3: Display the transaction details for one transaction id using record (table-based)

Ans:

```
DECLARE
Transaction_rec Transaction%rowtype;
BEGIN
SELECT * into Transaction_rec
FROM Transaction
WHERE T_id = 447;
dbms_output.put_line('Transaction ID: ' || Transaction_rec.T_id);
dbms_output.put_line('Transaction sales type: ' || Transaction_rec.Sales_Type);
dbms_output.put_line('Transaction manager name: ' || Transaction_rec.Manager_name);
dbms_output.put_line('Transaction qantity: ' || Transaction_rec.Quantity);
dbms_output.put_line('Transaction time and date: ' || Transaction_rec.Time_Date);
dbms_output.put_line('Transaction player id: ' || Transaction_rec.Player_id);
END;
/
```

Cursor

Q1: Display a player name and its jersey number using cursor (one row)

Ans:

```
declare
P_name Player.Player_name%type;
P_jersye Player.Jersey%type;
cursor p_player is
select Player_name,Jersey from Player;
begin
open p_Player;
fetch p_Player into P_name ,P_jersey ;
dbms_output.put_line(P_name||' '||P_jersey );
close p_Player;
end
/
```

Q2: Display all player name and their address using cursor(multiple row)

Ans:

```
declare
player_name Player.p_name%type;
Player_address Player.p_address%type;
cursor p_Player is
select p_name,p_address from Player;
```

```
begin

open p_Player;

loop

fetch p_Player into player_name,player_address;

exit when p_player%notfound;

dbms_output.put_line(player_name||' '||Player_address);

end loop;

close p_Player;

end

/
```

Q3: Update player value using cursor (implicit cursor attributes)

Ans:

```
DECLARE var_rows number(5);

BEGIN

UPDATE Player Value

SET price = price + ;

IF SQL%NOTFOUND THEN

dbms_output.put_line('None of the price were updated');

ELSE IF SQL%FOUND THEN

var_rows := SQL%ROWCOUNT;

dbms_output.put_line('Price for ' || var_rows || 'Items are updated');

END IF;

END;
```

Trigger

Q1: Create the Category trigger and update it

Ans:

```
CREATE OR REPLACE TRIGGER Category_added

after INSERT ON Category

FOR EACH ROW

BEGIN

dbms_output.put_line('New Category Added');

END;

select * from Category;

insert into Category values ('Neymar','10');

rollback
```

Q2: Create the Player_ID trigger and use (BEFORE UPDATE)

Ans:

```
CREATE or REPLACE TRIGGER Player_ID_Update

BEFORE UPDATE ON Player_ID

FOR EACH ROW

Begin

dbms_output.put_line('Id updated');

END;
```

Q3: Create a row level trigger for the Transaction table that would fire for DELETE operations performed on the Transaction table.

Ans:

```
CREATE OR REPLACE TRIGGER display_Quantity_changes

BEFORE DELETE ON Transaction

FOR EACH ROW

DECLARE

quantity_diff number;

BEGIN

quantity_diff := :NEW.Player- :OLD.Player;

dbms_output.put_line('Old Player: ' || :OLD.Player);

dbms_output.put_line('New Player: ' || :NEW.Player);

dbms_output.put_line('Player difference: ' || player_diff );

END;

/
```

Package

Q1: Create a package, which can display player name.

Ans:

```
CREATE PACKAGE Player_pack AS

PROCEDURE display_name(p_Id

Player.Id%type);

END Player_pack ;

/
```

Q2: Create a package body, which can display player name.

Ans:

```
CREATE or Replace PACKAGE BODY Player_pack AS

PROCEDURE display_name(P_id Player.Id%TYPE) IS

P_name Player.Player_name%TYPE;

BEGIN

SELECT Player_name INTO P_name

FROM Player

WHERE Id= P_id ;

dbms_output.put_line('Player Name: '|| P_name );

END display_name;

END Player_pack ;

/
```

Q3: Display Player name for a Player id using package.

Ans:

```
begin

Player_pack.display_name('16');

End
```

CONCLUSION

Finally, we create a management system that can help a club in various way. Club management store their data easily and safely. Club fan know about the club in more way. Players and manager see their stats. We need to work in this management system. Need to insert more players data, manager data and staff data. We also need to add more members so that it looks like a proper sports club management system.