AMERICAN INTERNATIONAL UNIVERSITY BANGLADESH (AIUB)

FACULTY OF SCIENCE & TECHNOLOGY

Course Title

INTRODUCTION TO DATABASE (2108)

Semester: Spring 23-24

Section: [L]

TITLE

Football Management System

Supervised By

MD Sajid Bin Faisal

Submitted By: Group no: 08

Name	ID
Nabil Mohammed Nasim Uddin	23-51206-1
Nasir Sarkar	22-48370-3
Nafisul Hasan Bhuiyan	22-49784-3
S. M. Sayed Al Habib	22-48365-3

TABLE OF CONTENTS

1. Introduction

Football is one of the most popular sports in the world. So, it is also a challenging thing to manage this sport. This project comes with a modern way of managing and storing the information of the entities named coach, team, match, referee, player and manager. MySQL, oracle, draw.io, XAMPP, MySQL java connector (Jar) and IDE have been used to create this project. It will make the football management system much easier for the organizer. Organizer will be beneficial by using it.

2. Case Study / Scenario

In a football management system, A team participate in various competitions, playing numerous matches throughout the season. Each match is held at a specific venue and is uniquely identified by a match_id. A team is represented by its t_id and t_name with each team having its own set of players, coaches, and manager. Each match is officiated by a single referee, who is identified by referee r_id, r_name and r_sal. Coaches are essential figures in team management and are identified by c_id, c_name, and c-sal. Players form the core of each team, bringing their unique talents and abilities to the field. Each player is identified by player p_id, p_name and p_sal. They undergo rigorous training and practice sessions to enhance their performance on the field. In every team, there is a designated manager who oversees the strategic aspects of team operations. The manager plays a pivotal role in decision-making, team coordination, and resource management. Managers are identified by m_id and m_name.

StudentID1: 22-49784-3 Name: Nafisul Hasan Bhuiyan	StudentID3: 23-51206-1 Name: Nabil Mohammed Nasim Uddin
StudentID2: 22-48370-3 Name: Nasir Sarkar	StudentID4: 22-48365-3 Name: S. M. Sayed Al Habib
CO2: Understand the fundamental concepts underlying database systems and gain hands-on experience with ER diagram Case study	
PO-c2: Develop process for complex computer science and engineering problems considering cultural and societal factors.	Marks

3. ER Diagram

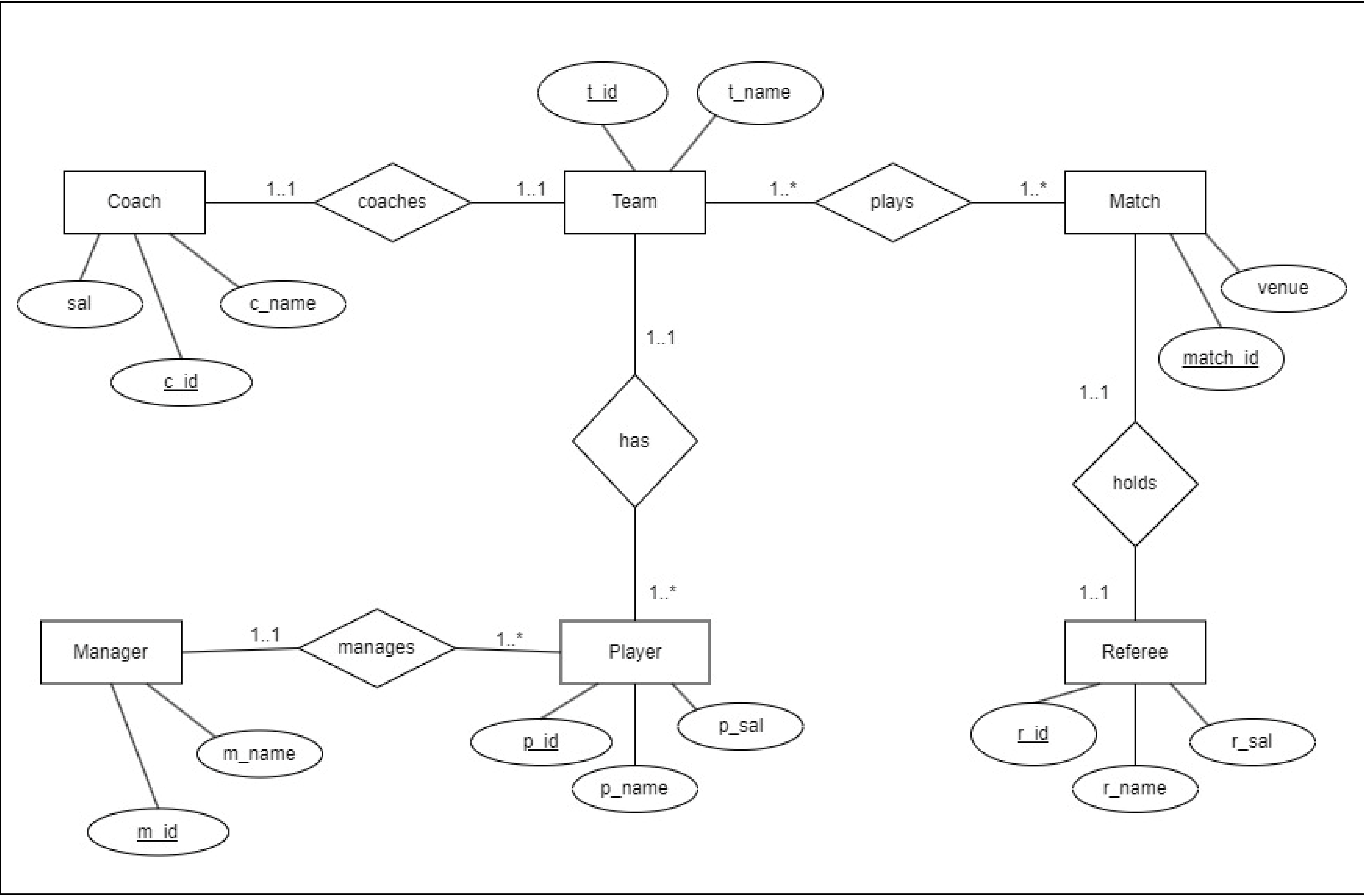


Fig-3.1: ER diagram of a football management system according to the case study

4. Normalization

4.1 Holds:

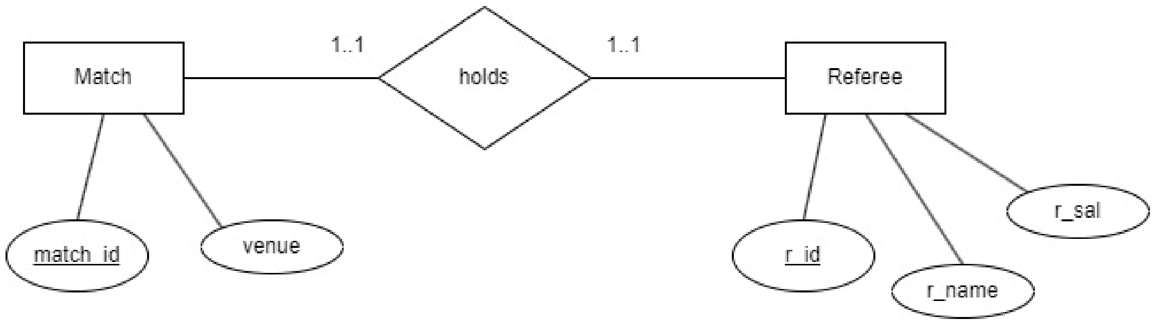


Fig-4.1: Shows the ‘holds’ relationship between Match and Referee

UNF: venue, <u>match_id</u>, <u>r_id</u>, r_sal, r_name

1NF: venue, <u>match_id</u>, <u>r_id</u>, r_sal, r_name

2NF:

1. venue, <u>match_id</u>
2. r_sal, r_name, r_id
3. <u>match_id</u> (Pk), r_id (Fk)

3NF:

1. venue, <u>match_id</u>
2. r_sal, r_name, r_id
3. <u>match_id</u> (Pk), r_id (Fk)

4.2 Manages:

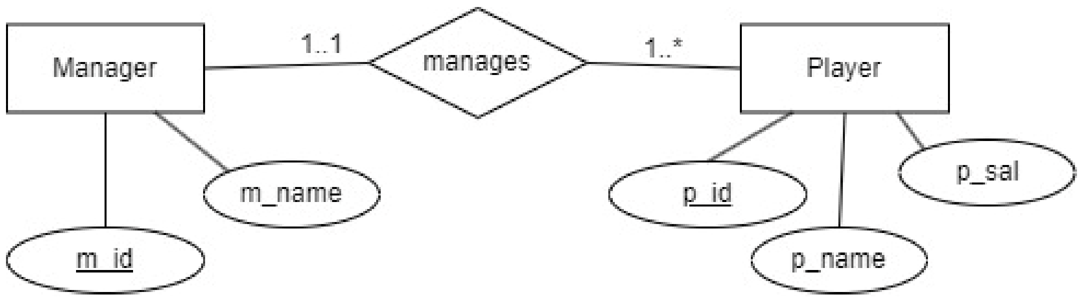


Fig-4.2: Shows the ‘manages’ relationship between Manager and Player

UNF: m_id, m_name, p_id, p_name, p_sal

1NF: m_id, m_name, p_id, p_name, p_sal

2NF:

1. m_id, m_name
2. p_id (Pk), p_name, p_sal, m_id (Fk)

3NF:

1. m_id, m_name
2. p_id (Pk), p_name, p_sal, m_id (Fk)

4.3 Has:

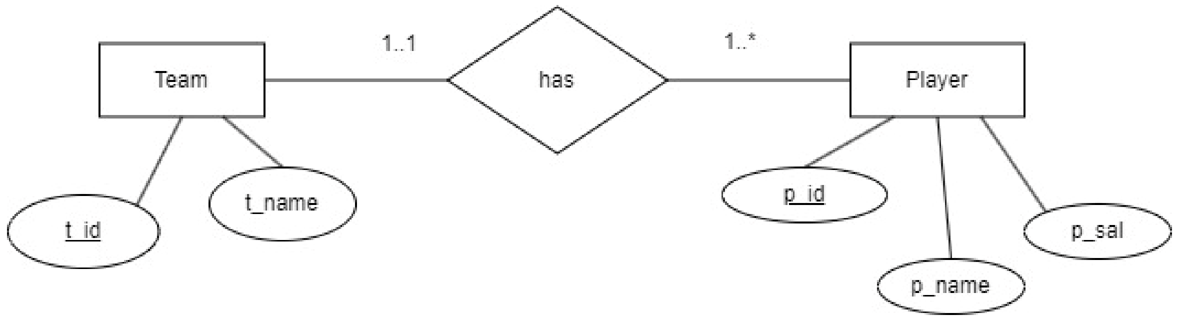


Fig-4.3: Shows the ‘has’ relationship between Team and Player

UNF: t_id, t_name, p_id, p_name, p_sal

1NF: t_id, t_name, p_id, p_name, p_sal

2NF:

1. t_id, t_name
2. p_id (Pk), p_name, p_sal, t_id (Fk)

3NF:

1. t_id, t_name
2. p_id (Pk), p_name, p_sal, t_id (Fk)

4.4 Plays:

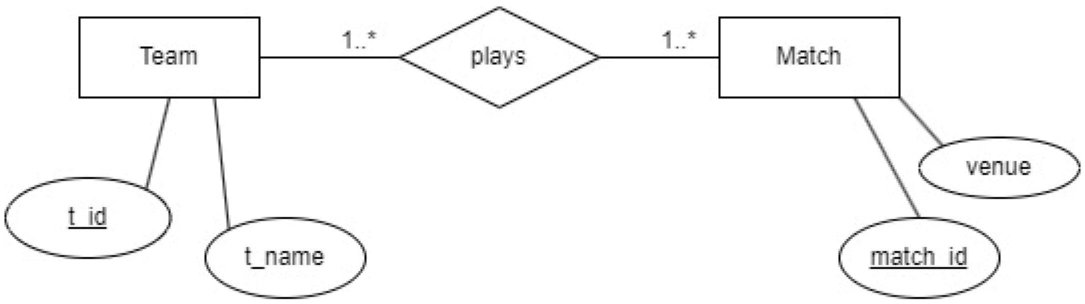


Fig-4.4: Shows the ‘plays’ relationship between Team and Match

UNF: t_id, t_name, match_id, venue

1NF: t_id, t_name, match_id, venue

2NF:

1. t_id, t_name
2. match_id, venue
3. t_id (Pk), match_id (Fk)

3NF:

1. t_id, t_name
2. match_id, venue
3. t_id (Pk), match_id (Fk)

4.5 Coaches:

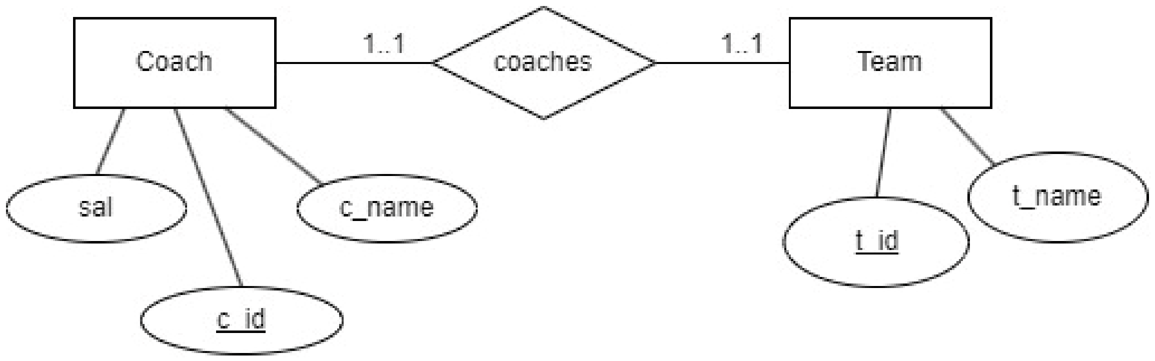


Fig-4.5: Shows the 'coaches' relationship between Coach and Team

UNF: sal, c_id, c_name, t_id, t_name

1NF: sal, c-id, c-name, t_id, t_name

2NF:

1. sal, c_id, c_name
2. t_id, t_name
3. c_id (Pk), t_id (Fk)

3NF:

1. sal, c_id, c_name
2. t_id, t_name
3. c_id (Pk), t_id (Fk)

5. Finalization

1. venue, match_id
2. r_sal, r_name, r_id
3. match_id (Pk), r_id (Fk)
4. m_id, m_name
5. p_id (Pk), p_name, p_sal, m_id (Fk)
6. t_id, t_name
7. p_id (Pk), p_name, p_sal, t_id (Fk)
8. t_id, t_name
9. match_id, venue
10. t_id (Pk), match_id (Fk)
11. sal, c-id, c_name
12. t_id, t_name
13. c_id (Pk), t-id (Fk)

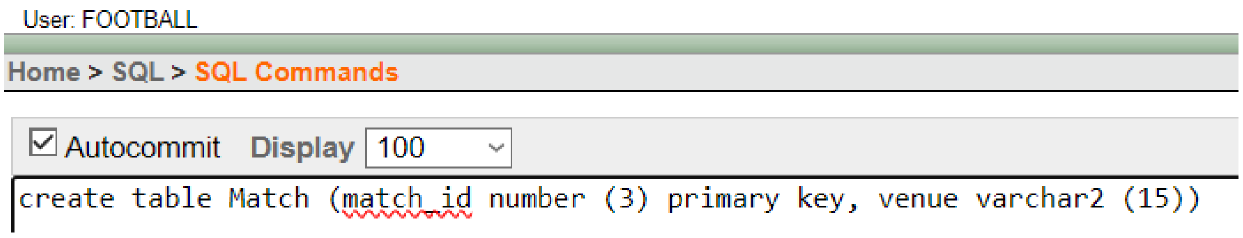
Final Table:

1. **Match:** venue, match_id
2. **Referee:** r_sal, r_name, r_id
3. **Holds:** match_id (Pk), r_id (Fk)
4. **Manager:** m_id, m_name
5. **Manages:** p_id (Pk), p_name, p_sal, m_id (Fk)
6. **Team:** t_id, t_name
7. **Has:** p_id (Pk), p_name, p_sal, t_id (Fk)
8. **Plays:** t_id (Pk), match_id (Fk)
9. **Coach:** sal, c-id, c_name
10. **Coaches:** c_id (Pk), t-id (Fk)

6. Table Creation (DDL Operations)

StudentID1: 22-49784-3 Name: Nafisul Hasan Bhuiyan	StudentID3: 23-51206-1 Name: Nabil Mohammed Nasim Uddin
StudentID2: 22-48370-3 Name: Nasir Sarkar	StudentID4: 22-48365-3 Name: S. M. Sayed Al Habib
CO4: Creating DML, DDL using Oracle and connection with ODBC/JDBC for existing JAVA application	
PO-e-2: Use modern engineering and IT tools for prediction and modeling of complex computer science and engineering problem	Marks

6.1 Match:



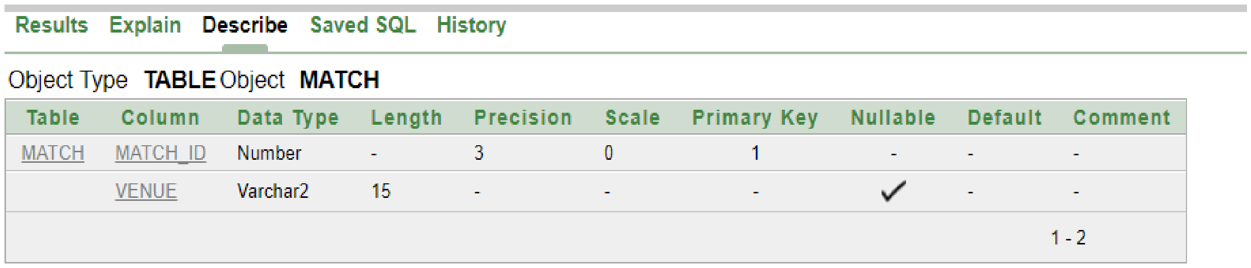
User: FOOTBALL

Home > SQL > SQL Commands

Autocommit Display 100

```
create table Match (match_id number (3) primary key, venue varchar2 (15))
```

Fig-6.1.1: Shows the query for creating table Match



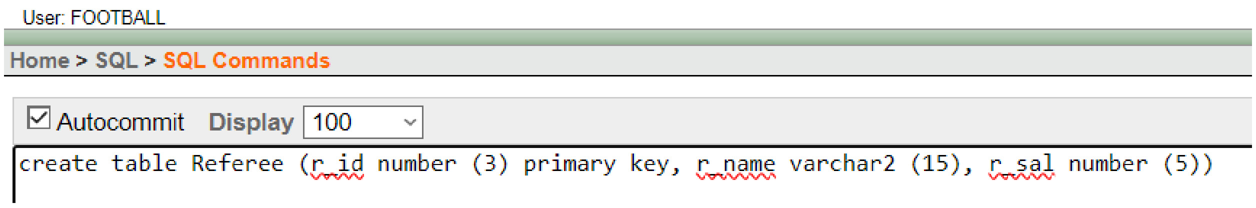
Object Type **TABLE** Object **MATCH**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MATCH	MATCH_ID	Number	-	3	0	1	-	-	-
	VENUE	Varchar2	15	-	-	-	✓	-	-

1 - 2

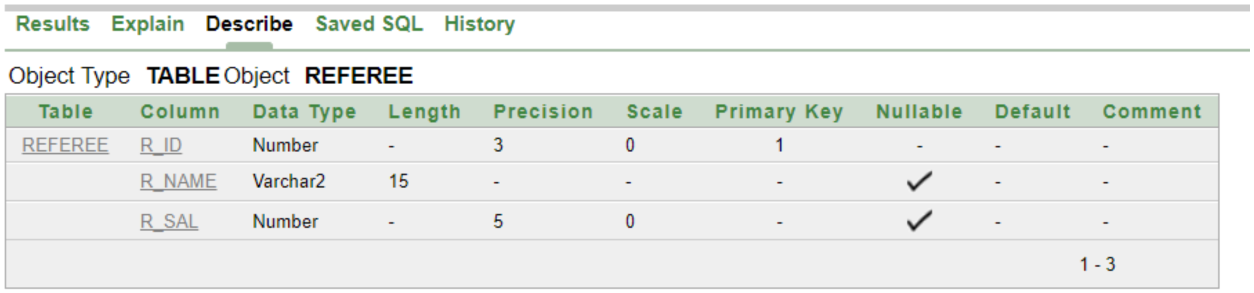
Fig-6.1.2: Shows the description of table Match

6.2 Referee:



```
create table Referee (r_id number (3) primary key, r_name varchar2 (15), r_sal number (5))
```

Fig-6.2.1: Shows the query for creating table Referee



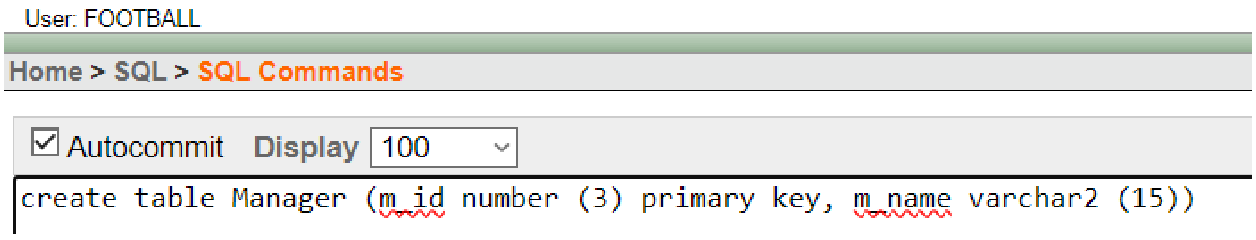
Object Type TABLE Object REFEREE

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
REFEREE	R_ID	Number	-	3	0	1	-	-	-
	R_NAME	Varchar2	15	-	-	-	✓	-	-
	R_SAL	Number	-	5	0	-	✓	-	-

1 - 3

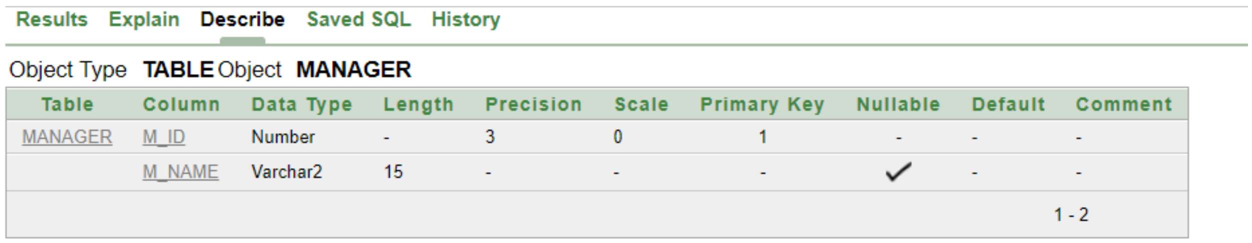
Fig-6.2.2: Shows the description of table Referee

6.3 Manager:



```
create table Manager (m_id number (3) primary key, m_name varchar2 (15))
```

Fig-6.3.1: Shows the query for creating table Manager



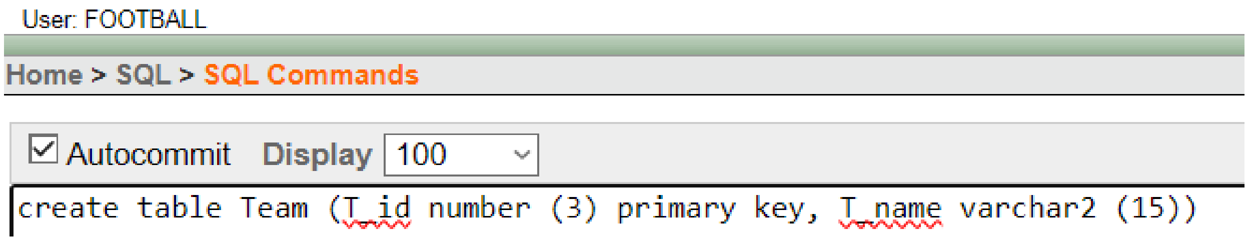
Object Type TABLE Object MANAGER

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MANAGER	M_ID	Number	-	3	0	1	-	-	-
	M_NAME	Varchar2	15	-	-	-	✓	-	-

1 - 2

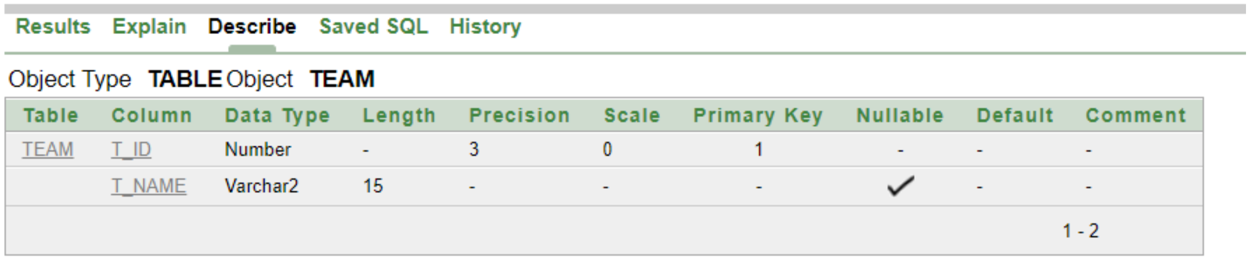
Fig-6.3.2: Shows the description of table Manager

6.4 Team:



```
create table Team (T_id number (3) primary key, T_name varchar2 (15))
```

Fig-6.4.1: Shows the query for creating table Team



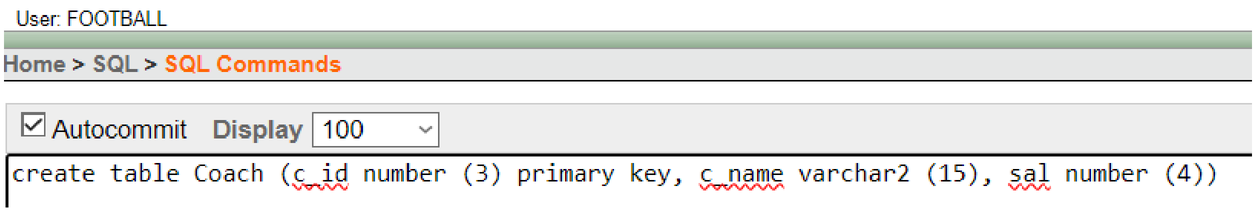
Object Type TABLE Object TEAM

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
TEAM	T_ID	Number	-	3	0	1	-	-	-
	T_NAME	Varchar2	15	-	-	-	✓	-	-

1 - 2

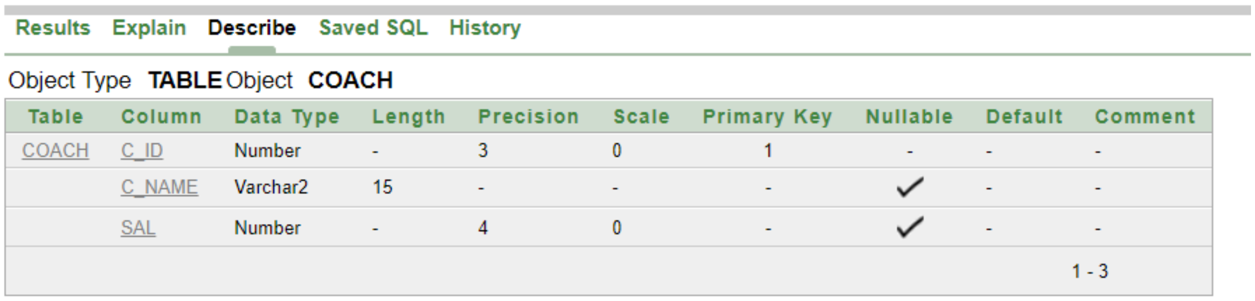
Fig-6.4.2: Shows the description of table Team

6.5 Coach:



```
create table Coach (c_id number (3) primary key, c_name varchar2 (15), sal number (4))
```

Fig-6.5.1: Shows the query for creating table Coach



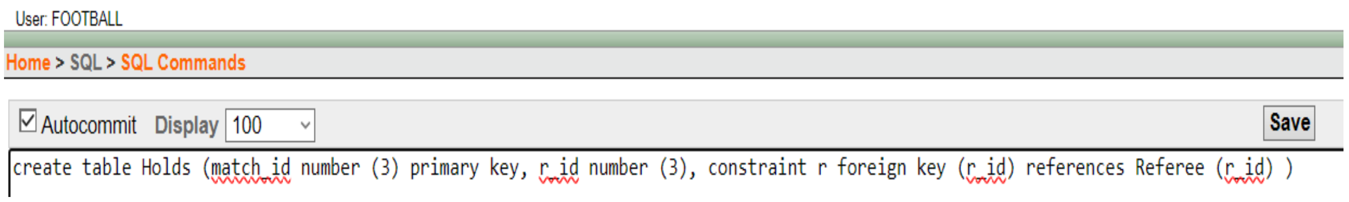
Object Type TABLE Object COACH

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
COACH	C_ID	Number	-	3	0	1	-	-	-
	C_NAME	Varchar2	15	-	-	-	✓	-	-
	SAL	Number	-	4	0	-	✓	-	-

1 - 3

Fig-6.5.2: Shows the description of table Coach

6.6 Holds:

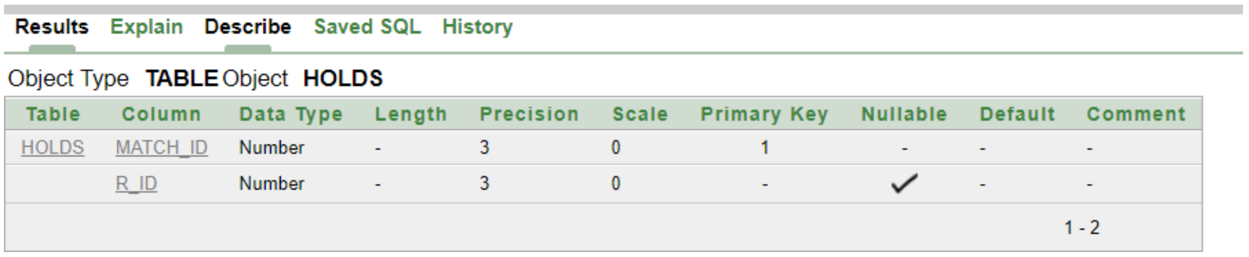

User: FOOTBALL

Home > SQL > SQL Commands

Autocommit Display 100 Save

```
create table Holds (match_id number (3) primary key, r_id number (3), constraint r foreign key (r_id) references Referee (r_id) )
```

Fig-6.6.1: Shows the query for creating table Holds

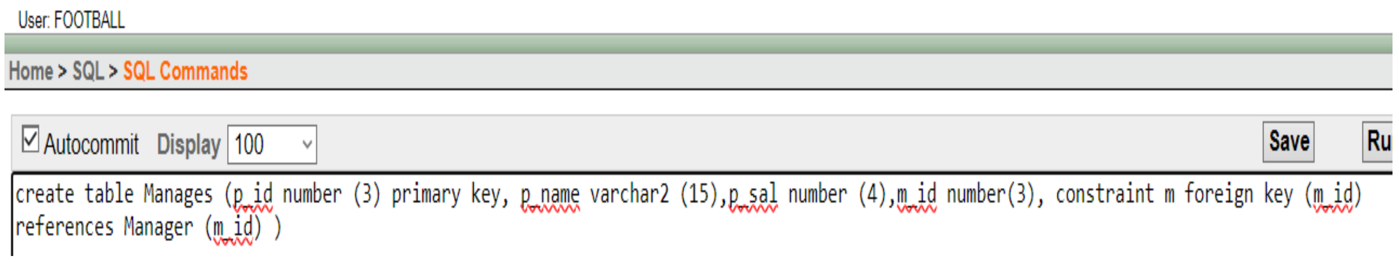

Results Explain Describe Saved SQL History

Object Type TABLE Object HOLDS

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
HOLDS	MATCH_ID	Number	-	3	0	1	-	-	-
	R_ID	Number	-	3	0	-	✓	-	-
									1 - 2

Fig-6.6.2: Shows the description of table Holds

6.7 Manages:

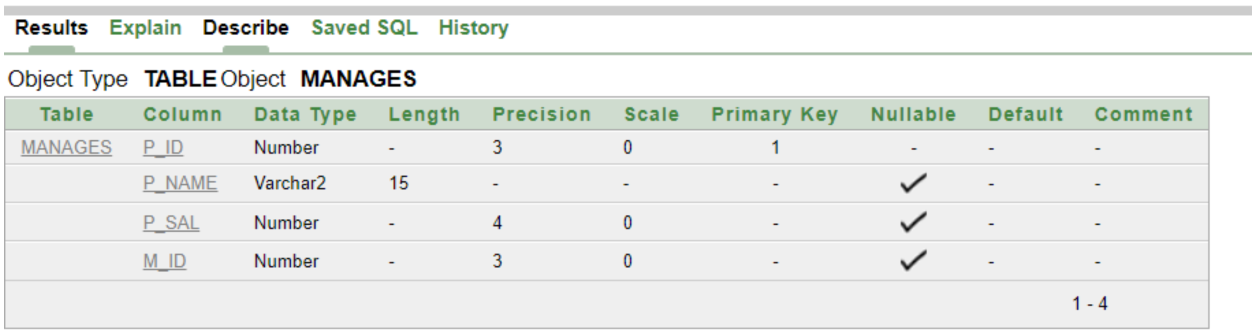

User: FOOTBALL

Home > SQL > SQL Commands

Autocommit Display 100 Save Ru

```
create table Manages (p_id number (3) primary key, p_name varchar2 (15),p_sal number (4),m_id number(3), constraint m foreign key (m_id)
references Manager (m_id) )
```

Fig-6.7.1: Shows the query for creating table Manages

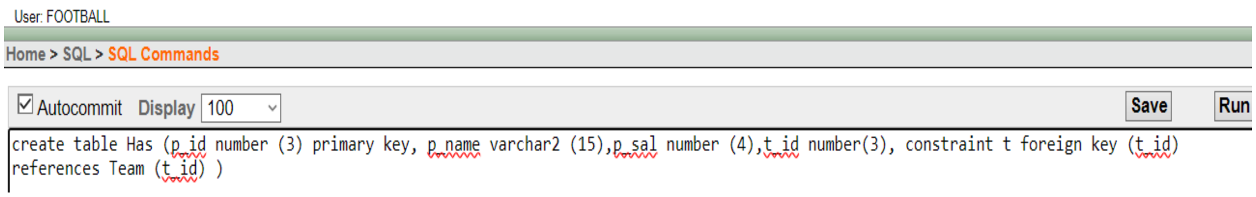

Results Explain Describe Saved SQL History

Object Type TABLE Object MANAGES

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MANAGES	P_ID	Number	-	3	0	1	-	-	-
	P_NAME	Varchar2	15	-	-	-	✓	-	-
	P_SAL	Number	-	4	0	-	✓	-	-
	M_ID	Number	-	3	0	-	✓	-	-
									1 - 4

Fig-6.7.2 Shows the description of table Manages

6.8 Has:

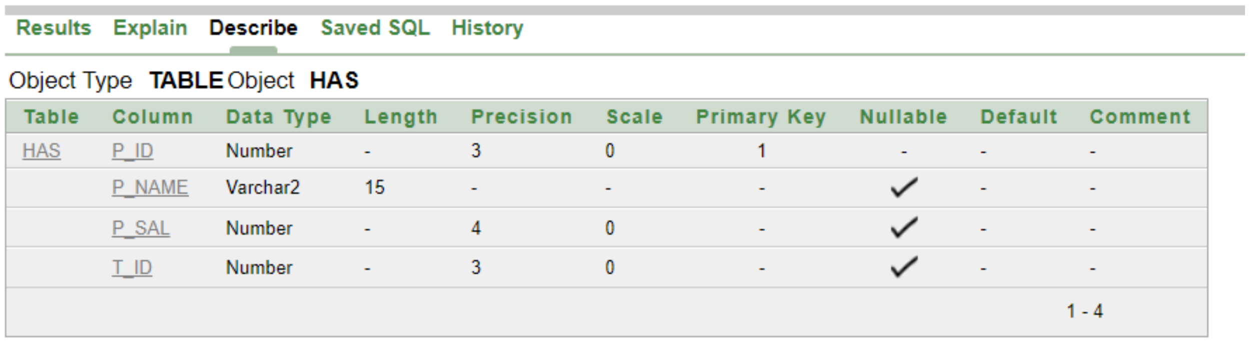

User: FOOTBALL

Home > SQL > SQL Commands

☑ Autocommit Display 100

```
create table Has (p_id number (3) primary key, p_name varchar2 (15),p_sal number (4),t_id number(3), constraint t foreign key (t_id)
references Team (t_id) )
```

Fig-6.8.1: Shows the query for creating table Has


Results Explain Describe Saved SQL History

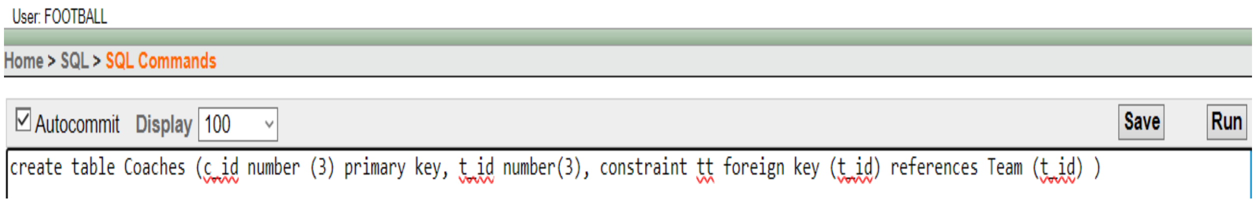
Object Type TABLE Object HAS

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
HAS	P_ID	Number	-	3	0	1	-	-	-
	P_NAME	Varchar2	15	-	-	-	✓	-	-
	P_SAL	Number	-	4	0	-	✓	-	-
	T_ID	Number	-	3	0	-	✓	-	-

1 - 4

Fig-6.8.2: Shows the description of table Has

6.9 Coaches:

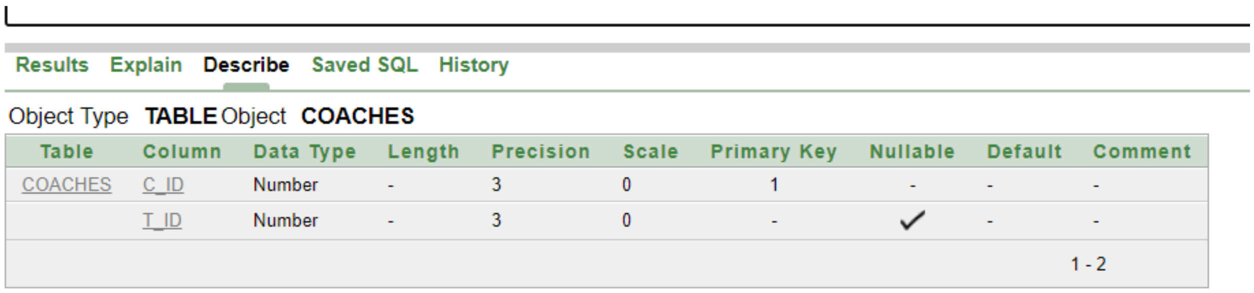

User: FOOTBALL

Home > SQL > SQL Commands

☑ Autocommit Display 100

```
create table Coaches (c_id number (3) primary key, t_id number(3), constraint tt foreign key (t_id) references Team (t_id) )
```

Fig-6.9.1: Shows the query for creating table Coaches


Results Explain Describe Saved SQL History

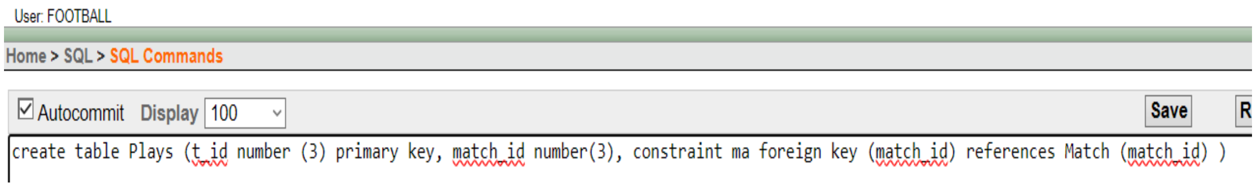
Object Type TABLE Object COACHES

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
COACHES	C_ID	Number	-	3	0	1	-	-	-
	T_ID	Number	-	3	0	-	✓	-	-

1 - 2

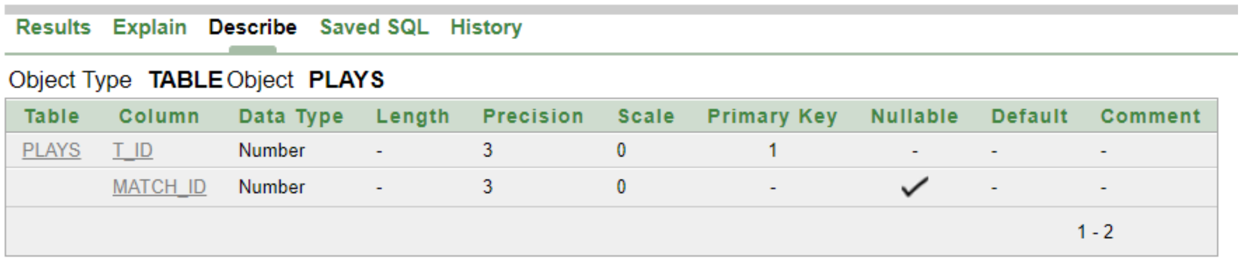
Fig-6.9.2: Shows the description of table Coaches

6.10 Plays:



```
create table Plays (t_id number (3) primary key, match_id number(3), constraint ma foreign key (match_id) references Match (match_id) )
```

Fig-6.10.1: Shows the query for creating table Plays



Object Type **TABLE** Object **PLAYS**

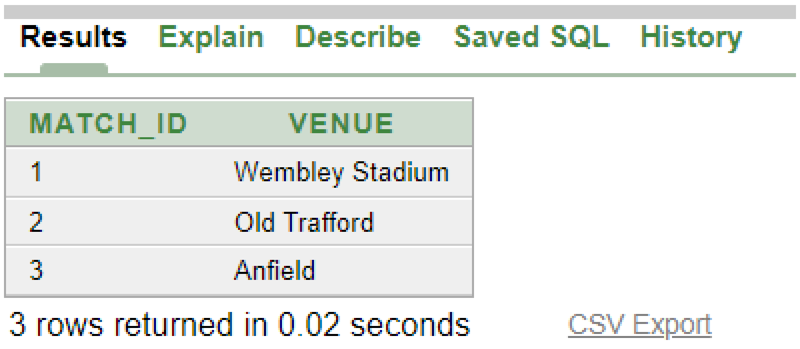
Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PLAYS	T_ID	Number	-	3	0	1	-	-	-
	MATCH_ID	Number	-	3	0	-	✓	-	-

1 - 2

Fig-6.10.2: Shows the description of table Plays

7. Inserted Values in the tables

7.1 Match:

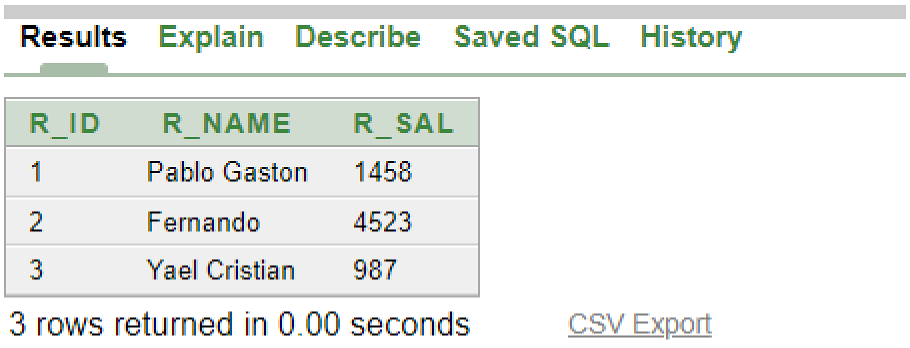


MATCH_ID	VENUE
1	Wembley Stadium
2	Old Trafford
3	Anfield

3 rows returned in 0.02 seconds

Fig-7.1: Shows the Values inserted for the table Match

7.2 Referee:

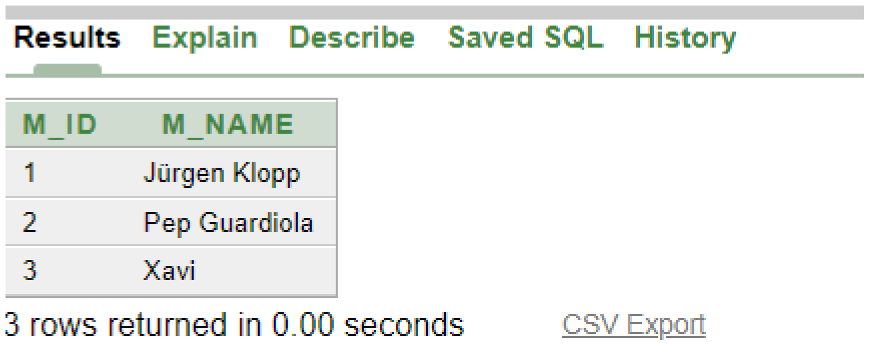


R_ID	R_NAME	R_SAL
1	Pablo Gaston	1458
2	Fernando	4523
3	Yael Cristian	987

3 rows returned in 0.00 seconds

Fig-7.2: Shows the Values inserted for the table Referee

7.3 Manager:

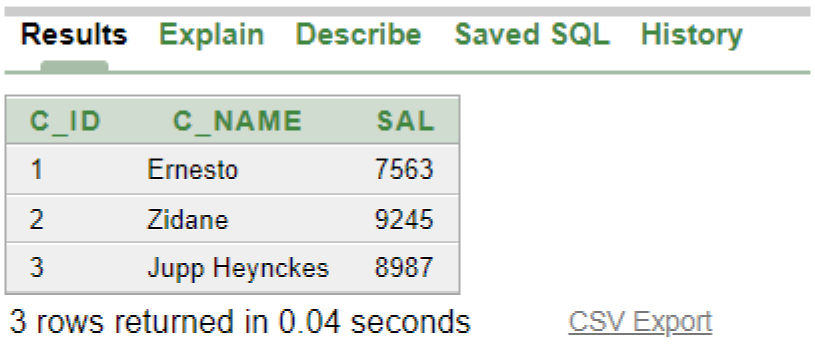


M_ID	M_NAME
1	Jürgen Klopp
2	Pep Guardiola
3	Xavi

3 rows returned in 0.00 seconds

Fig-7.3: Shows the Values inserted for the table Manager

7.4 Coach:

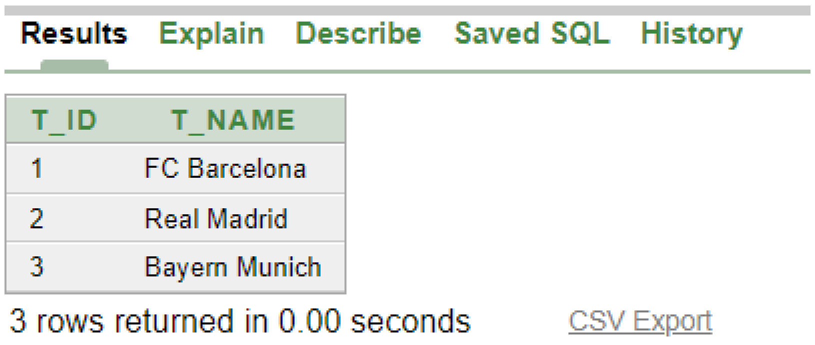


C_ID	C_NAME	SAL
1	Ernesto	7563
2	Zidane	9245
3	Jupp Heynckes	8987

3 rows returned in 0.04 seconds

Fig-7.4: Shows the Values inserted for the table Coach

7.5 Team:

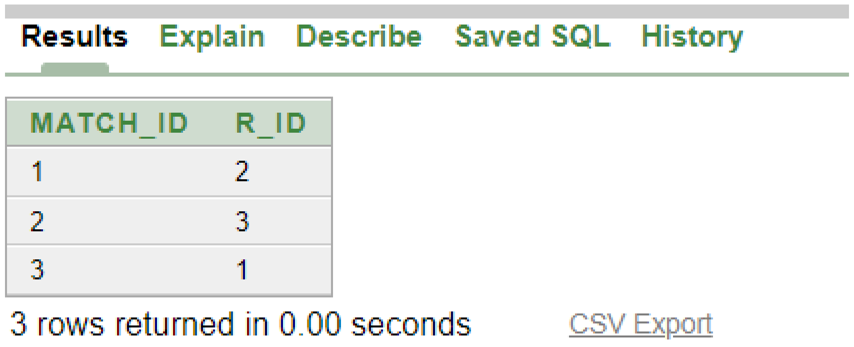


T_ID	T_NAME
1	FC Barcelona
2	Real Madrid
3	Bayern Munich

3 rows returned in 0.00 seconds

Fig-7.5: Shows the Values inserted for the table Team

7.6 Holds:

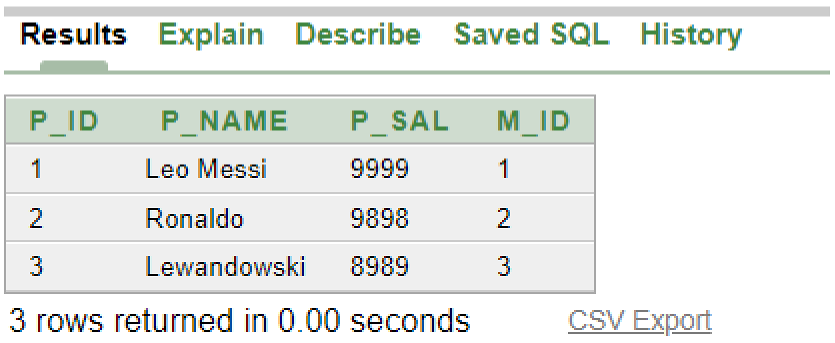


MATCH_ID	R_ID
1	2
2	3
3	1

3 rows returned in 0.00 seconds

Fig-7.6: Shows the Values inserted for the table Holds

7.7 Manages:

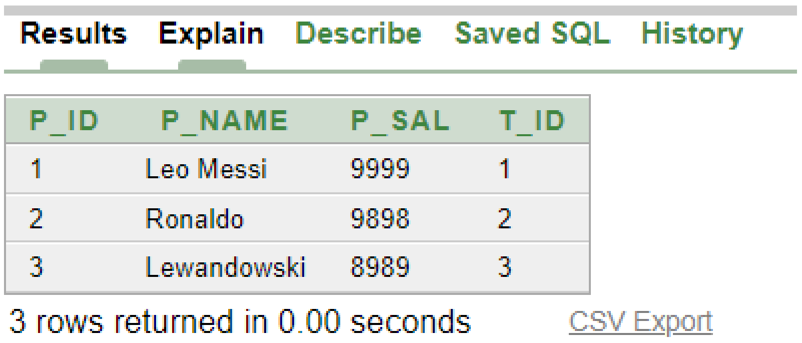

Results Explain Describe Saved SQL History

P_ID	P_NAME	P_SAL	M_ID
1	Leo Messi	9999	1
2	Ronaldo	9898	2
3	Lewandowski	8989	3

3 rows returned in 0.00 seconds CSV Export

Fig-7.7: Shows the Values inserted for the table Manages

7.8 Has:

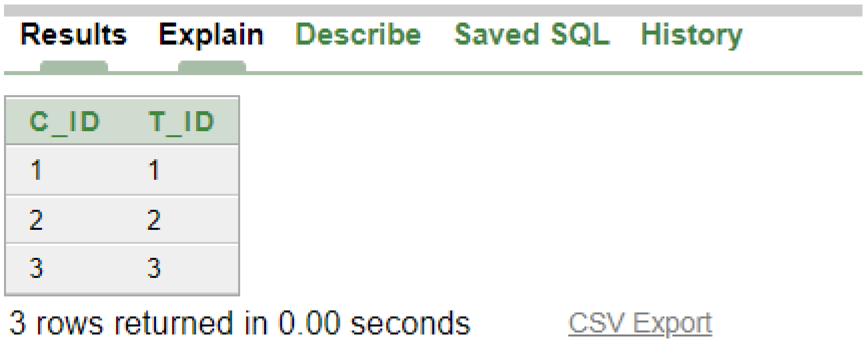

Results Explain Describe Saved SQL History

P_ID	P_NAME	P_SAL	T_ID
1	Leo Messi	9999	1
2	Ronaldo	9898	2
3	Lewandowski	8989	3

3 rows returned in 0.00 seconds CSV Export

Fig-7.8: Shows the Values inserted for the table Has

7.9 Coaches:

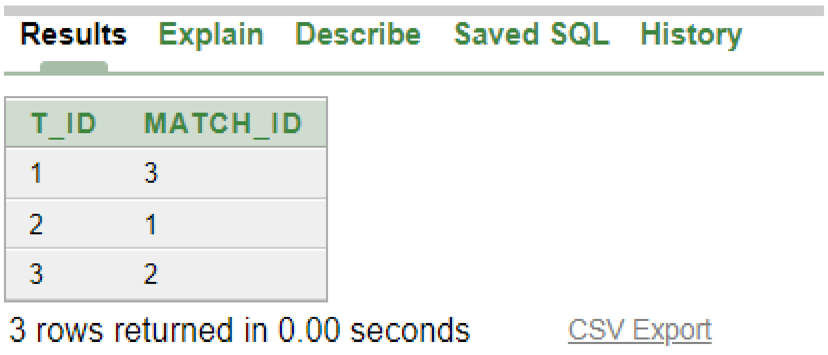

Results Explain Describe Saved SQL History

C_ID	T_ID
1	1
2	2
3	3

3 rows returned in 0.00 seconds CSV Export

Fig-7.9: Shows the Values inserted for the table Coaches

7.10 Plays:



T_ID	MATCH_ID
1	3
2	1
3	2

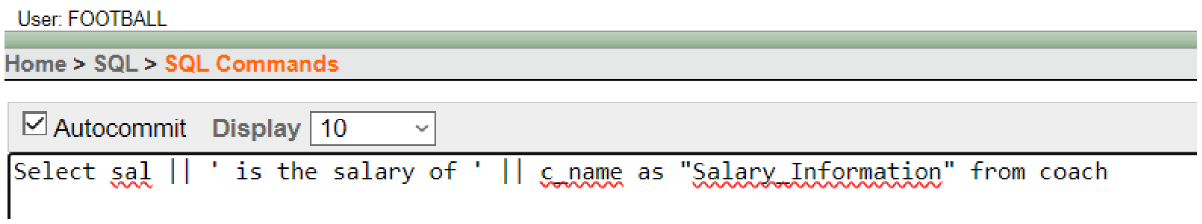
3 rows returned in 0.00 seconds

Fig-7.10: Shows the Values inserted for the table Plays

8. Query Test in DB

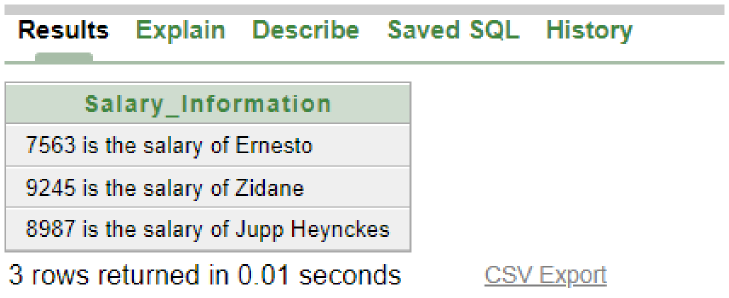
8.1 Simple Query:

Q: Show the salary of each coach like ex: 123 is the salary of xyz as salary info.



```
Select sal || ' is the salary of ' || c_name as "Salary_Information" from coach
```

Fig-8.1.1: Shows the command of the simple Query



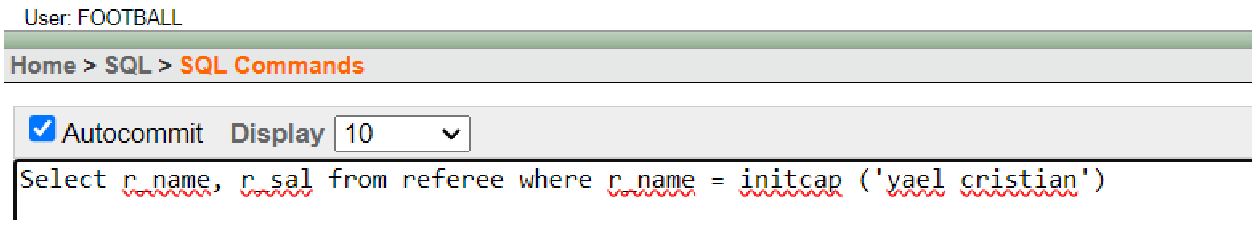
Salary_Information
7563 is the salary of Ernesto
9245 is the salary of Zidane
8987 is the salary of Jupp Heynckes

3 rows returned in 0.01 seconds

Fig-8.1.2: Shows the result of simple Query

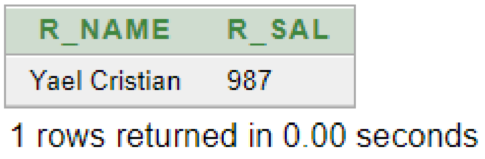
8.2 Single Row Function:

Q1: Show the name and salary of the referee whose name is ‘yael cristian’.



```
Select r_name, r_sal from referee where r_name = initcap ('yael cristian')
```

Fig-8.2.1.1: Shows the command of the single row function Query

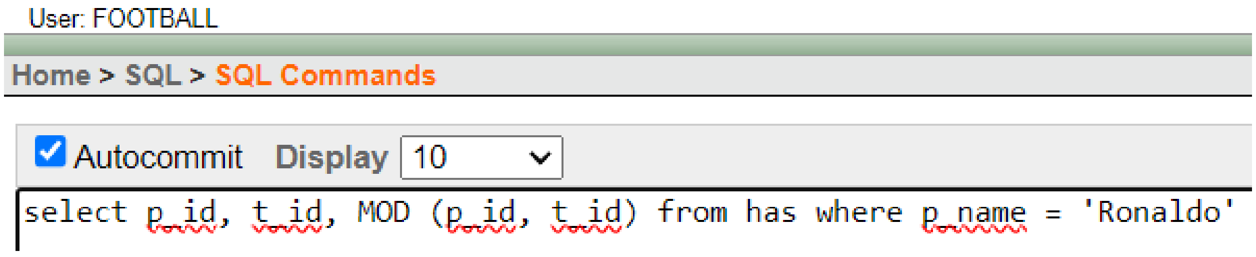


R_NAME	R_SAL
Yael Cristian	987

1 rows returned in 0.00 seconds

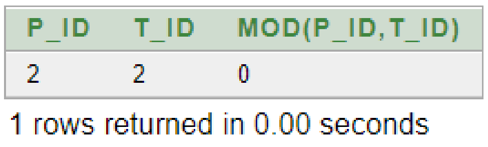
Fig-8.2.1.2: Shows the result of the single row function Query

Q2: Show player id, team id and also mod of these where player is Ronaldo.



```
select p_id, t_id, MOD (p_id, t_id) from has where p_name = 'Ronaldo'
```

Fig-8.2.2.1: Shows the command of the single row function Query



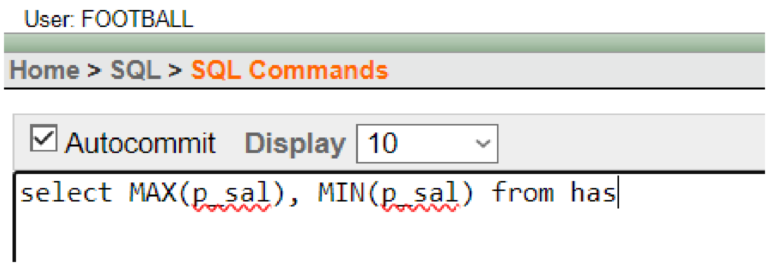
P_ID	T_ID	MOD(P_ID,T_ID)
2	2	0

1 rows returned in 0.00 seconds

Fig-8.2.2.2: Shows the result of the single row function Query

8.3 Aggregate Function:

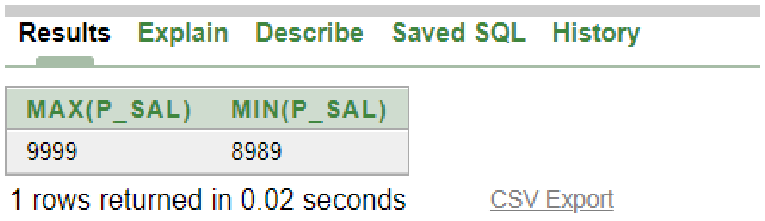
Q: Find the max player salary and the minimum player salary.



User: FOOTBALL

```
select MAX(p_sal), MIN(p_sal) from has
```

Fig-8.3.1: Shows the command of the aggregate function Query



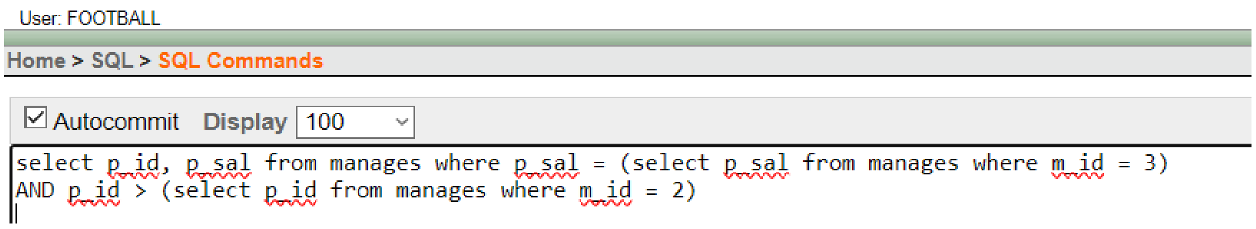
```
MAX(P_SAL)   MIN(P_SAL)
9999         8989

1 rows returned in 0.02 seconds
```

Fig-8.3.2: Shows the result of the aggregate function Query

8.4 Subquery:

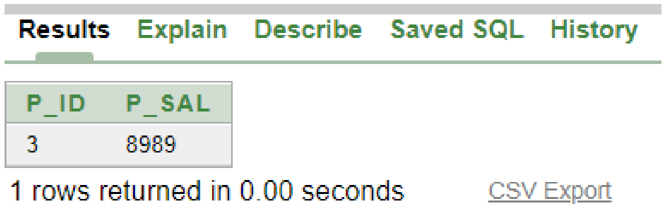
8.4.1 Single Row Subquery:

Q: Show the player id and player salary where the salary of the player is same as the salary of the player who is managed by m_id 3 and the player id should be greater than the player id of the player managed by m_id 2.



```
select p_id, p_sal from manages where p_sal = (select p_sal from manages where m_id = 3)
AND p_id > (select p_id from manages where m_id = 2)
```

Fig-8.4.1.1: Shows the command of the single row Subquery.



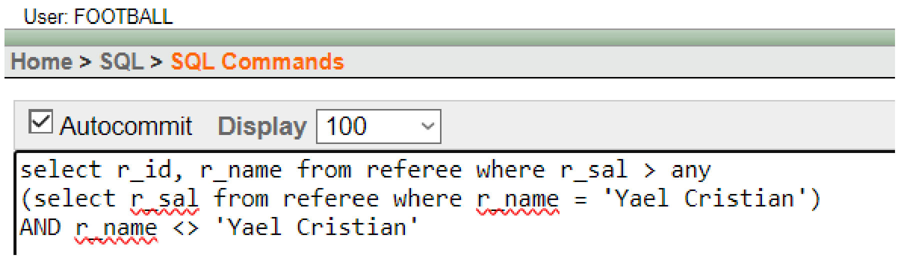
P_ID	P_SAL
3	8989

1 rows returned in 0.00 seconds

Fig-8.4.1.2: Shows the result of the single row Subquery.

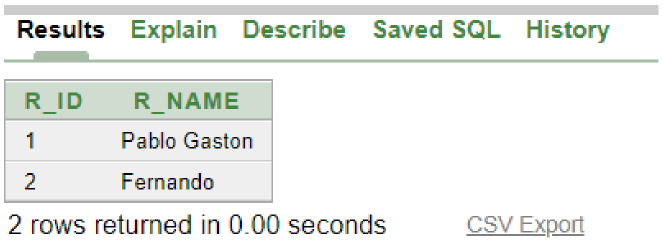
8.4.2 Multiple Row Subquery:

Q: Show the name and salary of referees whose salary is greater than any Yael Cristian named referee and whose name is not Yael Cristian.



```
select r_id, r_name from referee where r_sal > any
(select r_sal from referee where r_name = 'Yael Cristian')
AND r_name <> 'Yael Cristian'
```

Fig-8.4.2.1: Shows the command of the Multiple row Subquery.



R_ID	R_NAME
1	Pablo Gaston
2	Fernando

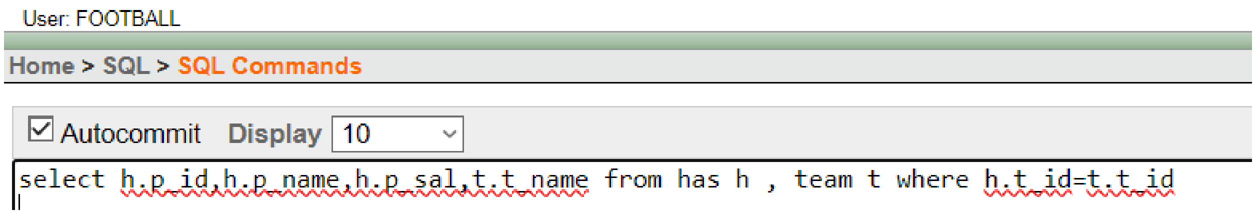
2 rows returned in 0.00 seconds

Fig-8.4.2.2: Shows the result of the multiple row Subquery

8.5 Joining:

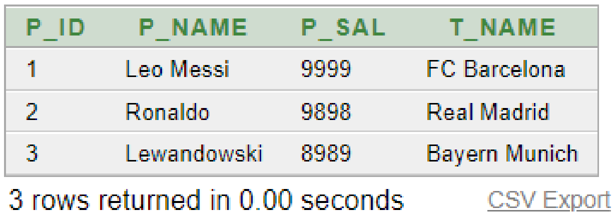
8.5.1 Equi-join:

Q: Show the player id, name, salary and also which team they are on.


User: FOOTBALL
Home > SQL > SQL Commands
Autocommit Display 10

```
select h.p_id,h.p_name,h.p_sal,t.t_name from has h , team t where h.t_id=t.t_id
```

Fig-8.5.1.1: Shows the command Query for Equi-join.



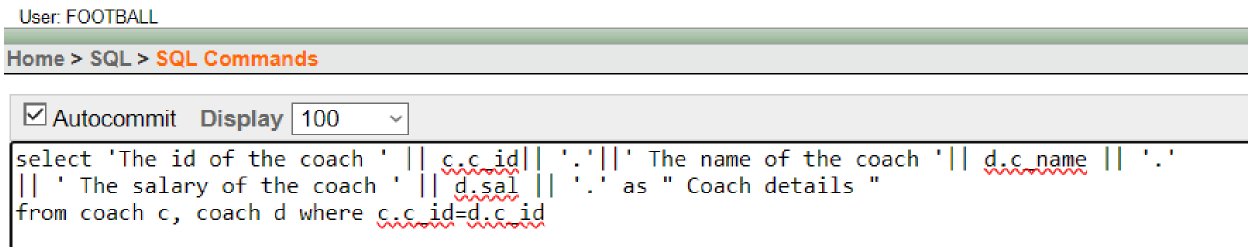
P_ID	P_NAME	P_SAL	T_NAME
1	Leo Messi	9999	FC Barcelona
2	Ronaldo	9898	Real Madrid
3	Lewandowski	8989	Bayern Munich

3 rows returned in 0.00 seconds CSV Export

Fig-8.5.1.2: Shows the Query test for Equi-join

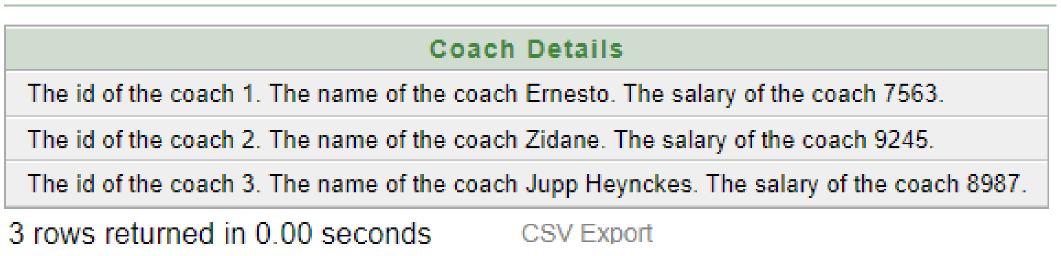
8.5.2 Self-join:

Q: Show the all details of coaches in the following way "The id of the coach 1. The name of the coach xyz. The salary of the coach 999."


User: FOOTBALL
Home > SQL > SQL Commands
Autocommit Display 100

```
select 'The id of the coach ' || c.c_id|| '.'||' The name of the coach '|| d.c_name || '.'
|| ' The salary of the coach ' || d.sal || '.' as " Coach details "
from coach c, coach d where c.c_id=d.c_id
```

Fig-8.5.2.1: Shows the command Query for Self-join.



Coach Details
The id of the coach 1. The name of the coach Ernesto. The salary of the coach 7563.
The id of the coach 2. The name of the coach Zidane. The salary of the coach 9245.
The id of the coach 3. The name of the coach Jupp Heynckes. The salary of the coach 8987.

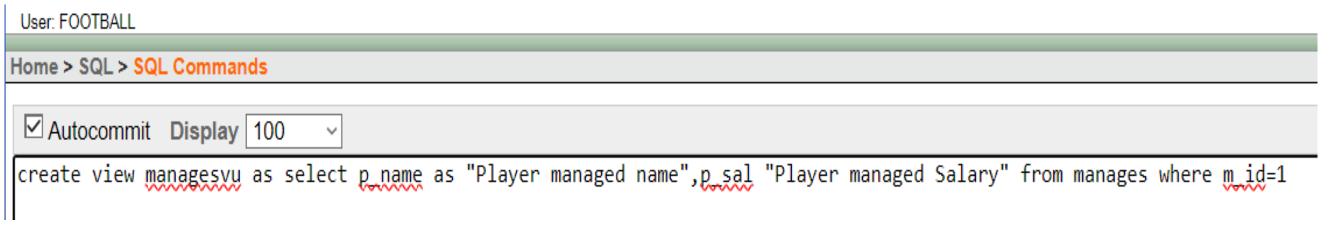
3 rows returned in 0.00 seconds CSV Export

Fig-8.5.2.2: Shows the Query test for Self-joining.

8.6 View:

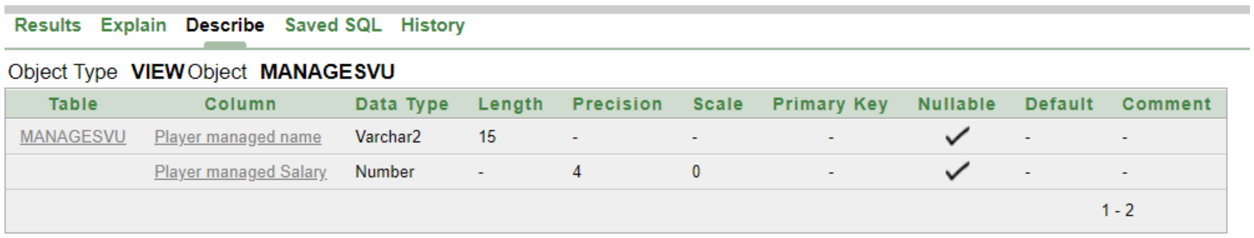
8.6.1 Simple View:

Q: Create view name managesvu where the m_id:1 shows p_name,p_sal that m_id: 1 manages



```
create view managesvu as select p_name as "Player managed name",p_sal "Player managed Salary" from manages where m_id=1
```

Fig-8.6.1.1: Shows the Query written to create simple view

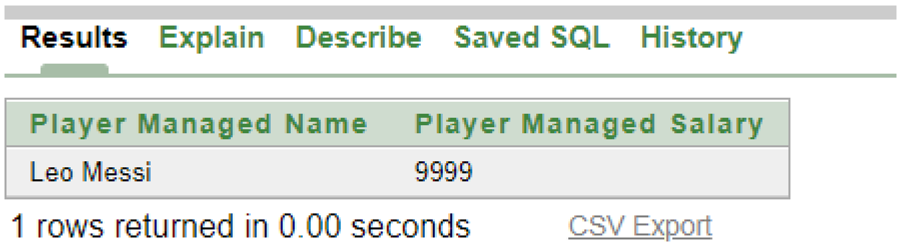


Object Type VIEW Object MANAGESVU

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MANAGESVU	Player managed name	Varchar2	15	-	-	-	✓	-	-
	Player managed Salary	Number	-	4	0	-	✓	-	-

1 - 2

Fig-8.6.1.2: Shows the description of the simple view created.



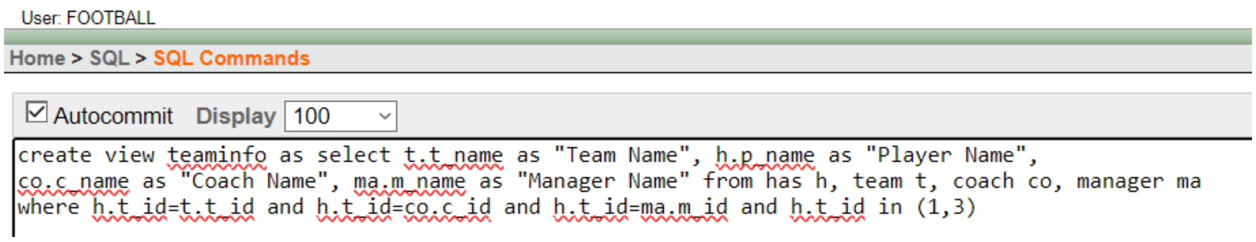
Player Managed Name	Player Managed Salary
Leo Messi	9999

1 rows returned in 0.00 seconds   CSV Export

Fig-8.6.1.3: Shows the details of the simple view Query

8.6.2 Complex View:

Q: Create a complex view name teaminfo where they will the then team name, player name coach name and manager name of the team whose ids are 1 and 3



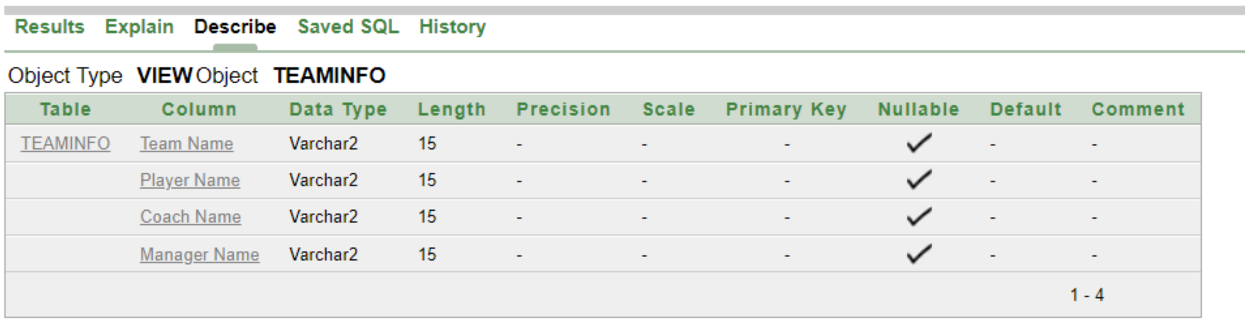
User: FOOTBALL

Home > SQL > SQL Commands

Autocommit Display 100

```
create view teaminfo as select t.t_name as "Team Name", h.p_name as "Player Name",
co.c_name as "Coach Name", ma.m_name as "Manager Name" from has h, team t, coach co, manager ma
where h.t_id=t.t_id and h.t_id=co.c_id and h.t_id=ma.m_id and h.t_id in (1,3)
```

Fig-8.6.2.1: Shows the Query written to create complex view

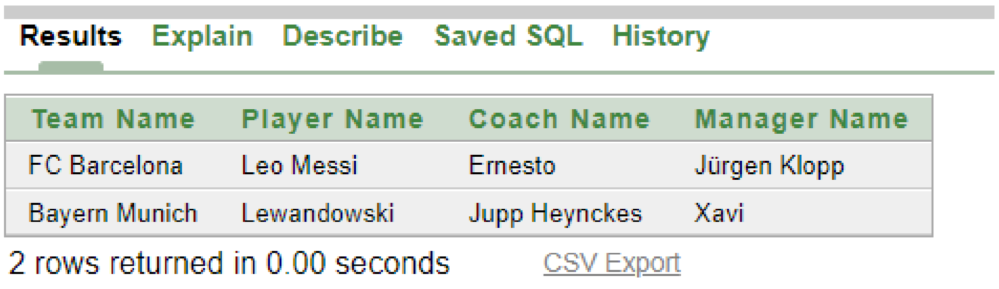


Object Type VIEW Object TEAMINFO

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
TEAMINFO	Team Name	Varchar2	15	-	-	-	✓	-	-
	Player Name	Varchar2	15	-	-	-	✓	-	-
	Coach Name	Varchar2	15	-	-	-	✓	-	-
	Manager Name	Varchar2	15	-	-	-	✓	-	-

1 - 4

Fig-8.6.2.2: Shows the description of the complex view created



Team Name	Player Name	Coach Name	Manager Name
FC Barcelona	Leo Messi	Ernesto	Jürgen Klopp
Bayern Munich	Lewandowski	Jupp Heynckes	Xavi

2 rows returned in 0.00 seconds CSV Export

Fig-8.6.2.3: Shows the details of the complex view Query

9. Database Connection

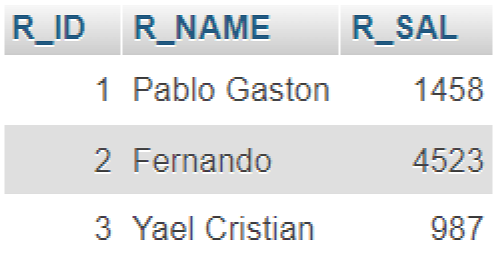
9.1 Name: Nasir Sarkar
ID: 22-48370-3
Table: Referee

Tools:

1. **MySQL Java Connector [Jar]:** To connect the database with the java code
2. **XAMPP:** To create the MySQL database in its server
3. **Visual Studio Code**: To write the java code and launch the whole program

Steps:

1. At first, I went to the control panel of **XAMPP**. After starting the **APACHE** and **MySQL**, I went to the **admin** option of MySQL. Then I created a database named **'football'**. Under this database I created a **'referee'** table and inserted values as shown in fig-9.1.1.

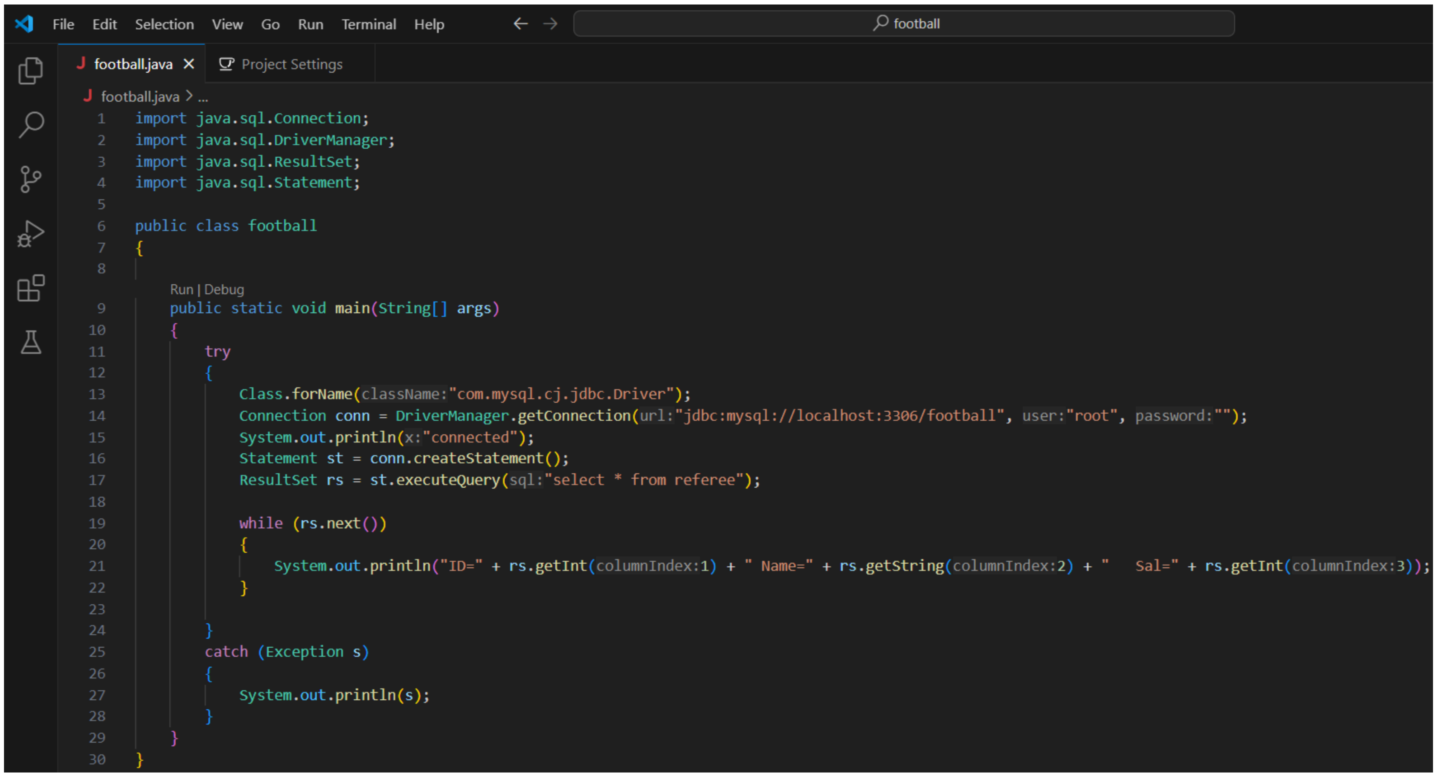


R_ID	R_NAME	R_SAL
1	Pablo Gaston	1458
2	Fernando	4523
3	Yael Cristian	987

Fig-9.1.1: Values of table created in MySQL through XAMPP

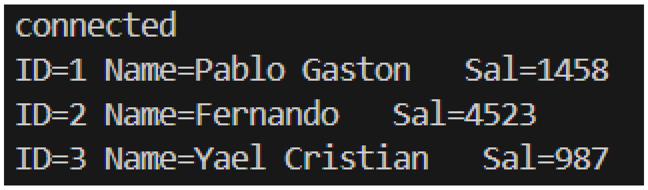
2. After the step1, I went to **Visual Studio Code** and created a project for java program. Then I added **MySQL Java Connector [Jar file]** as a library in that project.

3. After completing all the steps, I wrote the java code at my created project in **Visual Studio Code** to connect the database through MySQL java Connector. Finally, I launched the program.



```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class football
{

    public static void main(String[] args)
    {
        try
        {
            Class.forName(className:"com.mysql.cj.jdbc.Driver");
            Connection conn = DriverManager.getConnection(url:"jdbc:mysql://localhost:3306/football", user:"root", password:"");
            System.out.println(x:"connected");
            Statement st = conn.createStatement();
            ResultSet rs = st.executeQuery(sql:"select * from referee");

            while (rs.next())
            {
                System.out.println("ID=" + rs.getInt(columnIndex:1) + " Name=" + rs.getString(columnIndex:2) + "   Sal=" + rs.getInt(columnIndex:3));
            }

        }
        catch (Exception s)
        {
            System.out.println(s);
        }
    }
}
```

Fig-9.1.2: Java program in Visual Studio Code to connect the database



```
connected
ID=1 Name=Pablo Gaston   Sal=1458
ID=2 Name=Fernando   Sal=4523
ID=3 Name=Yael Cristian   Sal=987
```

Fig-9.1.3: Connected database info as output in Visual Studio Code

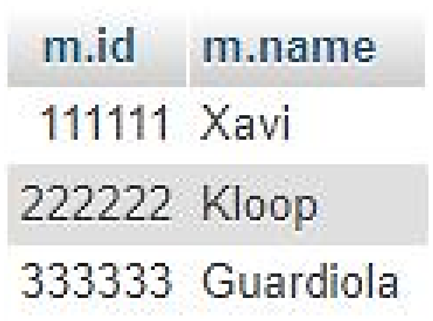
9.2 Name: Nafisul Hasan Bhuiyan
ID: 22-49784-3 (TABLE: MANAGER)

Steps of DB Connection:

1. Firstly, **mysql-connector-java version 8.0.28** was downloaded , **XAMPP** (**xampp apache mariadb perl php**) was downloaded and installed where servers can be created .

2. Secondly, I accessed the control panel of **XAMPP**. Following that, I initiated both **APACHE** and **MySQL** services. Subsequently, I navigated to the **MySQL** admin option.

3. Following Step **(2)** I created a database and named that **' Manager '** and inserted values depicted in figure 9.2.1 .



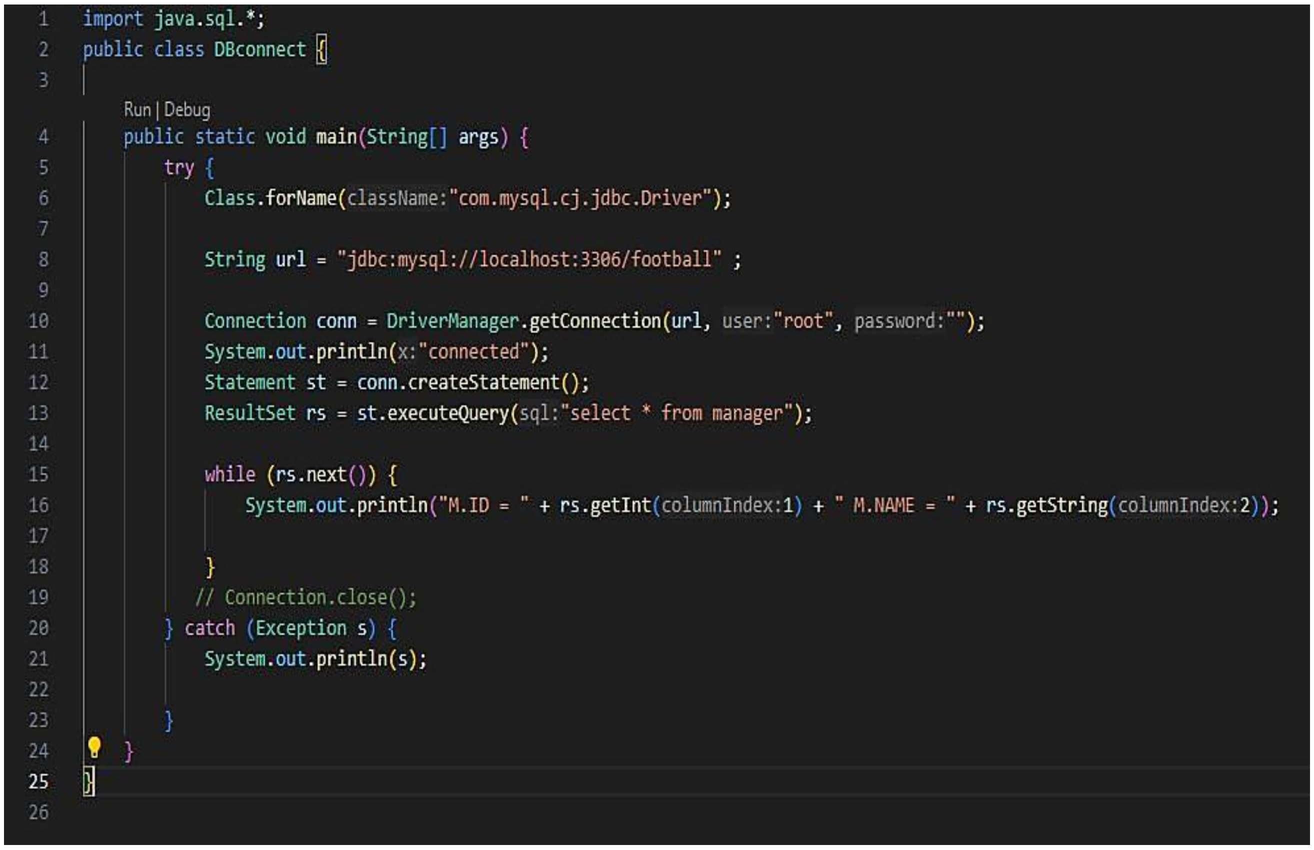
m.id	m.name
111111	Xavi
222222	Kloop
333333	Guardiola

Fig - 9.2.1: Values of Manager table from MySQL server

Connecting DB to Java Project :

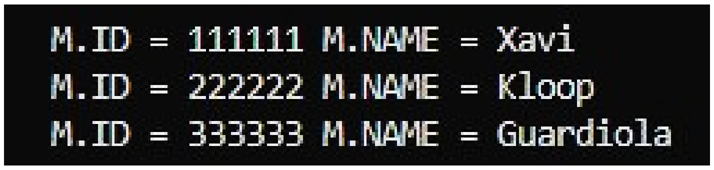
4. I used **Visual Studio Code** as my **IDE** and created a java project and named the file DBconnect.java .

5. Once all the steps were finished, I wrote the Java code within my Visual Studio project to establish a connection with the database . Ultimately, I executed the program. Showed in following Fig – 9.2.2.



```java
import java.sql.*;
public class DBconnect {

    Run | Debug
    public static void main(String[] args) {
        try {
            Class.forName(className:"com.mysql.cj.jdbc.Driver");

            String url = "jdbc:mysql://localhost:3306/football" ;

            Connection conn = DriverManager.getConnection(url, user:"root", password:"");
            System.out.println(x:"connected");
            Statement st = conn.createStatement();
            ResultSet rs = st.executeQuery(sql:"select * from manager");

            while (rs.next()) {
                System.out.println("M.ID = " + rs.getInt(columnIndex:1) + " M.NAME = " + rs.getString(columnIndex:2));

            }
           // Connection.close();
        } catch (Exception s) {
            System.out.println(s);

        }
    }
}
```

Fig – 9.2.2: Code that connected the database with java.



```
M.ID = 111111 M.NAME = Xavi
M.ID = 222222 M.NAME = Kloop
M.ID = 333333 M.NAME = Guardiola
```

Fig – 9.2.3: Output of the Code

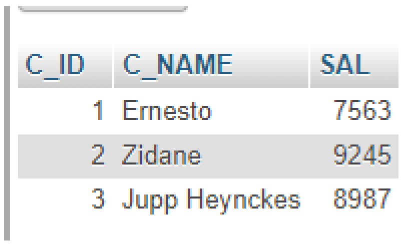
9.3 Name: Nabil Mohammed Nasim Uddin
ID: 23-51206-1

Step By Step Process:

1. First, I have installed XAMPP from **(XAMPP APACHE MARIADB PERL PHP).**

2. After completing the downloading process, I opened the XAMPP Control and started Apache and MySQL. Right after starting I pressed the admin button beside the start of MySQL.

3. It leads me to a page where I created a system named FOOTBALL and then proceeded to create a table named Coach and inserted the following Values.



C_ID	C_NAME	SAL
1	Ernesto	7563
2	Zidane	9245
3	Jupp Heynckes	8987

Fig.9.3.1 Shows the values inserted in the XAMPP server of the system Football and table Coach.

4. I chose and IDE which is visual studio code to connect my database. For connecting my database I needed a jar connector file which is Mysql java connector (jar file) and I downloaded this from (**mysql java connector maven)** and the version was 8.0.28.

5. Now I have connected the jar file with the library of the IDE the screenshot below shows the jar file being added to the library.

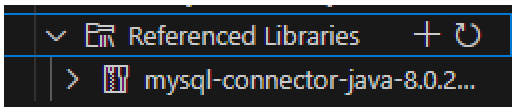
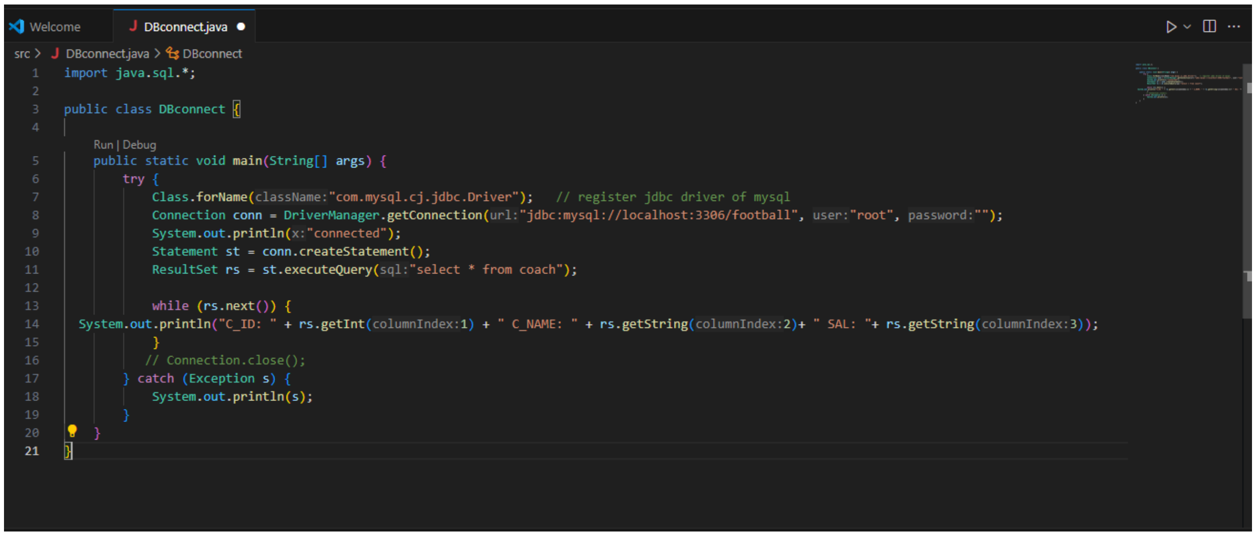


Fig.9.3.2 Shows the jar file added to the library.

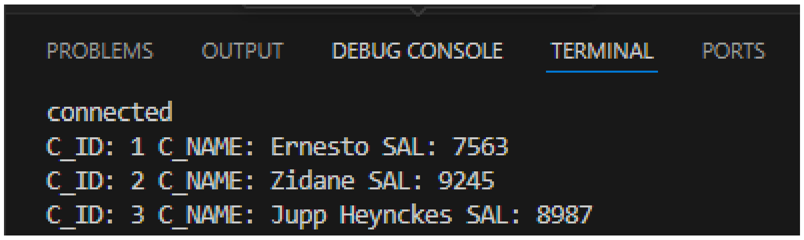
6. After adding the library to the file I then wrote a code to connect to the database which I created previously in XAMPP server. This code has been created solely for connecting my database with the XAMPP server. The picture below will show the code written to connect database table COACH.



```java
import java.sql.*;

public class DBconnect {

    public static void main(String[] args) {
        try {
            Class.forName(className:"com.mysql.cj.jdbc.Driver");   // register jdbc driver of mysql
            Connection conn = DriverManager.getConnection(url:"jdbc:mysql://localhost:3306/football", user:"root", password:"");
            System.out.println(x:"connected");
            Statement st = conn.createStatement();
            ResultSet rs = st.executeQuery(sql:"select * from coach");

            while (rs.next()) {
  System.out.println("C_ID: " + rs.getInt(columnIndex:1) + " C_NAME: " + rs.getString(columnIndex:2)+ " SAL: "+ rs.getString(columnIndex:3));
            }
           // Connection.close();
        } catch (Exception s) {
            System.out.println(s);
        }
    }
}
```

Fig.9.3.3. Shows the code written to connect database table Coach from XAMPP server.

7. The next picture will show the output of the following code. It will display connected at first and then the values of the table COACH. The picture below will show the output of the code.



```
connected
C_ID: 1 C_NAME: Ernesto SAL: 7563
C_ID: 2 C_NAME: Zidane SAL: 9245
C_ID: 3 C_NAME: Jupp Heynckes SAL: 8987
```

Fig.9.3.4. Shows the output of the code used to connect the database.

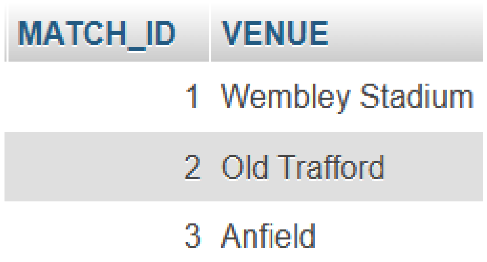
9.4 Name: S. M. Sayed Al Habib
ID: 22-48365-3 (Table Name: Match)

1. The mysql java connector(jar file) is installed at first . For that, the mysql java connector maven containing jar file of version 8.4.0 is downloaded.

2. The xampp apache mariadb perl php mysql xampp server id downloaded which contains many servers.Among that the apache and mysql server is started and admin panel of mysql server is opened

3. After that in that panel a database system is created namely “Footballmngt”,and a table namely “Match” which contains three columns . The subsequent values are then inserted into the table comprising four rows in total.

Here MySQL table-



MATCH_ID	VENUE
1	Wembley Stadium
2	Old Trafford
3	Anfield

Fig - 9.4.1: Output of the table from MySQL server.

4. Then an IDE is downloaded and installed namely “Apache Netbeans IDE20”.Inside it, the jar file is added in the library section of the new project created in the IDE.

5. After than a DB connection code is written iniside the “LMGT” class of the project. The basis of the code comprises of the following criteria’s-

➢ Register Driver- In the provided code, the line Class.forName(“com.mysql.cj.jdbc.Driver”); is used to dynamically load the MySQL JDBC (Java Database Connectivity) driver. In JDBC, drivers are used to establish a connection between a Java application and a database. Loading the driver is necessary to register it with the DriverManager, which allows the application to use the specified database driver.

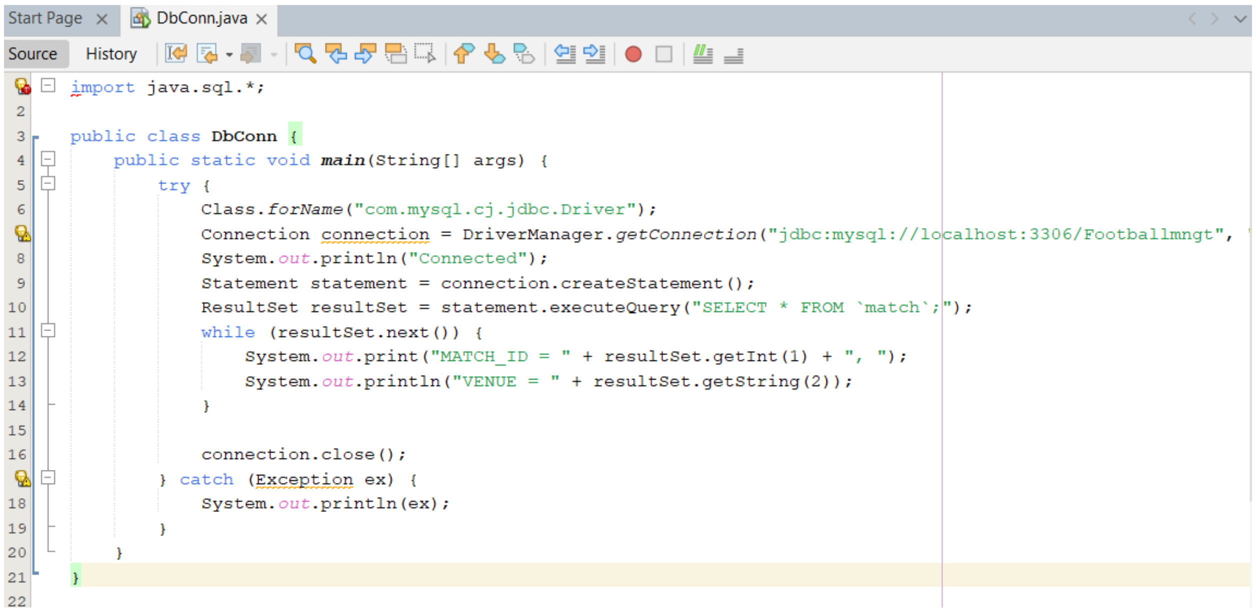
➢ Connection of DB- The DriverManager is used to establish a connection to a database by loading the appropriate driver and creating a connection.The connection interfeace represents a connection to a database.It provides methods for creating statements.The connection object is obtained from the DriverManager by calling the getConnection method with a URL,username and password.

➢ Statement- The Statement interface represents a SQL statement that can executed against a database.Statement objects are created using the createStatement method of a Connection

➢ Execution of the Query() in the Statement- The ResultSet interface represents the result set of a SQK query. It provides methods for retrieving data from the result set,iterating through rows, and accessing column values.ResultSet objects are obtained by executing a query on a Statement

➢ Connection close()-The connection.close() method in Java is used to close a database connection. Closing a connection is important for memory management, resource management because its important to release the resources when they are no longer needed.

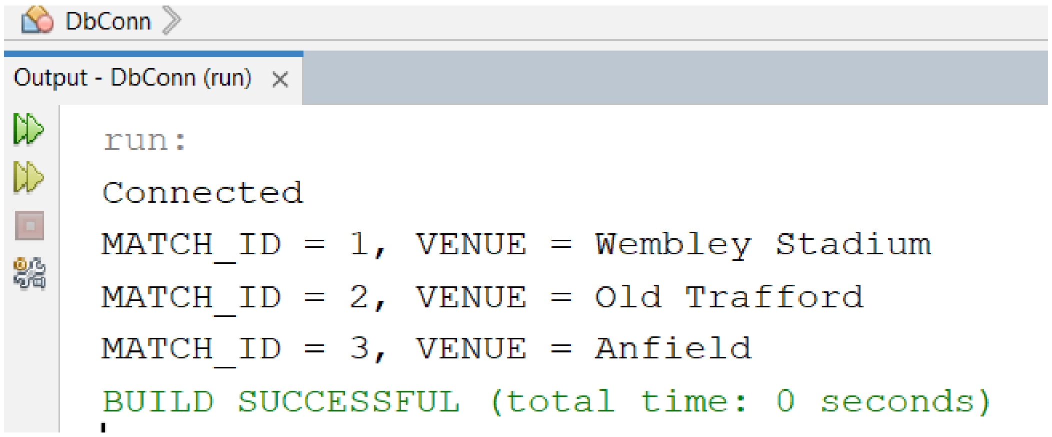
The main Code-



```java
import java.sql.*;

public class DbConn {
    public static void main(String[] args) {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/Footballmngt", '
            System.out.println("Connected");
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery("SELECT * FROM `match`;");
            while (resultSet.next()) {
                System.out.print("MATCH_ID = " + resultSet.getInt(1) + ", ");
                System.out.println("VENUE = " + resultSet.getString(2));
            }

            connection.close();
        } catch (Exception ex) {
            System.out.println(ex);
        }
    }
}
```

Fig – 9.4.2: Java code to connect MySQL database and to show the output.

The output-



```
run:
Connected
MATCH_ID = 1, VENUE = Wembley Stadium
MATCH_ID = 2, VENUE = Old Trafford
MATCH_ID = 3, VENUE = Anfield
BUILD SUCCESSFUL (total time: 0 seconds)
```

Fig – 9.4.3: Output of the query from the code.

Conclusion

In the Football Management System, the primary objective is to optimize the management processes of football. This comprehensive system typically comprises modules catering to various functions such as player management, match scheduling, venue analysis, and referee management. Additionally, it facilitates the generation of insightful reports and statistics to aid in strategic decision-making and future planning. By incorporating cutting-edge database management techniques, the system enables efficient data storage, retrieval, and manipulation, ensuring seamless operations and enhanced decision support. Furthermore, the implementation of advanced analytics tools empowers stakeholders to gain valuable insights into player performance, team dynamics, and fan engagement, thereby driving continual improvement and success on and off the field. Ultimately, the Football Management System aims to revolutionize the way football organizations operate, competitiveness, and sustainability in the world of sports.