

Laboratory 7

Title of the Laboratory Exercise: Nested queries and Join queries

1. Introduction and Purpose of Experiment

Nesting of queries within another one is known as a nested queries. The query within another is known as a subquery. The statement containing a subquery is called a Parent Statement. The parent statement uses the rows returned by the subquery. SQL Join is used for combining column from two or more tables by using values common to both tables. Join Keyword is used in SQL queries. By doing this lab, students will be able to implement nested queries and join queries.

2. Aim and Objectives

Aim

- To design and implement nested queries and join queries using SQL commands

Objectives

At the end of this lab, the student will be able to

- Design nested queries and join queries for the given problem statement
- Execute the nested queries and join queries

3. Experimental Procedure

- Analyse the problem statement
- Create tables with appropriate attributes
- Insert attribute values into the table
- Design nested queries and join queries
- Execute the SQL commands
- Test the executed commands
- Document the Results
- Analyse and discuss the outcomes of your experiment

4. Questions

- a. Create tables for the given relational schema. Assume appropriate data type, and key constraints for each field.

Player (Name, Id, TeamNo, Score)

Team (TeamNo, TeamName)

- b. Write the appropriate query for the following statements using SQL commands
- Find the names of all the players who are in the same Team of 'Smith' (use nested query)
 - Display the information about players who got Scores more than any player in TeamNo=1 (use nested query)
 - Display the players and Team details , in which the *TeamNo* is same in both the players and *Team* (without join)
 - Display the players and Team details , in which the *TeamNo* is same in both the players and *Team*
 - Display the players and Team details , in which the *TeamNo* is same in both the players and *Team* (use natural join)
 - Display the players and their team names, in which the *TeamNo* is same in both the players and *Team* (use left outer join)
 - Display the team names and the players involved, in which the *TeamNo* is same in both the players and *Team* (use right outer join)
- c. Create suitable front end for querying and displaying the results

5. Presentation of Results

Question A

MySQL Commands

```

1  -- CREATING TABLES AND DESCRIBING IT..
2
3  CREATE table Team(Team_No int,Team_Name VARCHAR(40),primary key(Team_No));
4  CREATE table Player(Player_ID int AUTO_INCREMENT,Name VARCHAR(40) Not NULL,Team_No int not null,Score int,
5  primary key(Player_ID), FOREIGN KEY (Team_No) REFERENCES Team(Team_No));
6
7  desc player;
8  desc team;

```

Figure 1 MySQL Command to Create TEAM and PLAYER TABLE

Result

```

mysql> desc team;
+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+
| Team_No    | int       | NO   | PRI | NULL    |       |
| Team_Name  | varchar(40) | YES  |     | NULL    |       |
+-----+
2 rows in set (0.00 sec)

mysql> desc player;
+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+
| Player_ID  | int       | NO   | PRI | NULL    | auto_increment |
| Name       | varchar(40) | NO   |     | NULL    |       |
| Team_No    | int       | NO   | MUL | NULL    |       |
| Score      | int       | YES  |     | NULL    |       |
+-----+
4 rows in set (0.01 sec)

```

Figure 2 Meta-Data of PLAYER and TEAM Table

Inserting data into tables

```

11  -- INSERTING DATA INTO TEAM TABLE
12
13  insert into team values (1001,"Bangalore");
14  insert into team values (1002,"Jaipur");
15  insert into team values (1003,"Delhi");
16  select * from team;
17

```

Figure 3 Insert Operation on TEAM Table

```

19  -- INSERTING DATA INTO PLAYER TABLE
20
21  insert into Player(Name,Team_No,Score) values ("K Srikanth",1001,4000);
22  insert into Player(Name,Team_No,Score) values ("P Supraja",1001,3000);
23  insert into Player(Name,Team_No,Score) values ("Bhoomika JS",1001,3000);
24  insert into Player(Name,Team_No,Score) values ("Apoorva R",1001,3000);
25  insert into Player(Name,Team_No,Score) values ("Naveen Kumar GN",1002,3000);
26  insert into Player(Name,Team_No,Score) values ("Sushanth",1003,3000);
27
28  select * from player;
29

```

Figure 4 Insert Operation on PLAYER Table

| Player_ID | Name | Team_No | Score |
|-----------|-----------------|---------|-------|
| 1 | K Srikanth | 1001 | 4000 |
| 2 | Naveen Kumar GN | 1002 | 3000 |
| 3 | Sushanth | 1003 | 3000 |
| 4 | P Supraja | 1001 | 3000 |
| 5 | Bhoomika JS | 1001 | 3000 |
| 6 | Apoorva R | 1001 | 3000 |

Figure 5 Select Operation on PLAYER Table

| Team_No | Team_Name |
|---------|-----------|
| 1001 | Bangalore |
| 1002 | Jaipur |
| 1003 | Delhi |

Figure 6 Select Operation on TEAM Table

Question B

- i. Find the names of all the players who are in the same Team of 'Smith' (use nested query)

MySQL Commands

```
30 -- Question 1
31
32 select Name,Team_No from player as p1 where p1.Team_No = (select Team_No from player as play where play.name = "K Srikanth");
```

Figure 7 MySQL Command for given problem statement Question 1

Result

| Name | Team_No |
|-------------|---------|
| K Srikanth | 1001 |
| P Supraja | 1001 |
| Bhoomika JS | 1001 |
| Apoorva R | 1001 |

Figure 8 MySQL Query result for given problem statement Question 1

- ii. Display the information about players who got Scores more than any player in TeamNo=1 (use nested query)

MySQL Commands

```
34 -- Question 2
35
36 select * from player as play where (play.score = (select max(score) from player as play2 where (play2.Team_No =1001)))
37
```

Figure 9 MySQL Command for given problem statement Question 2

Result

| Player_ID | Name | Team_No | Score |
|---------------|---------------|---------------|---------------|
| abc Filter... | abc Filter... | abc Filter... | abc Filter... |
| 1 | K Srikanth | 1001 | 4000 |

Figure 10 MySQL Query result for given problem statement Question 2

- iii. Display the players and Team details , in which the *TeamNo* is same in both the players and *Team* (without join)

MySQL Commands

```
38 -- Question 3
39
40 select * from player as play where (play.Team_No in (select Team_NO from team));
41
```

Figure 11 MySQL Command for given problem statement Question 3

Result

| Player_ID | Name | Team_No | Score |
|---------------|-----------------|---------------|---------------|
| abc Filter... | abc Filter... | abc Filter... | abc Filter... |
| 1 | K Srikanth | 1001 | 4000 |
| 2 | Naveen Kumar GN | 1002 | 3000 |
| 3 | Sushanth | 1003 | 3000 |
| 4 | P Supraja | 1001 | 3000 |
| 5 | Bhoomika JS | 1001 | 3000 |
| 6 | Apoorva R | 1001 | 3000 |

Figure 12 MySQL Query result for given problem statement Question 3

- iv. Display the players and Team details , in which the *TeamNo* is same in both the players and *Team*

MySQL Commands

```

42 -- Question 4
43
44 select * from player as play,team as tm where play.Team_No = tm.Team_No;
45

```

Figure 13 MySQL Command for given problem statement Question 4

Result

| Player_ID | Name | play.Team_No | Score | tm.Team_No | Team_Name |
|-----------|-----------------|--------------|-----------|------------|-----------|
| Filter... | Filter... | Filter... | Filter... | Filter... | Filter... |
| 1 | K Srikanth | 1001 | 4000 | 1001 | Bangalore |
| 2 | Naveen Kumar GN | 1002 | 3000 | 1002 | Jaipur |
| 3 | Sushanth | 1003 | 3000 | 1003 | Delhi |
| 4 | P Supraja | 1001 | 3000 | 1001 | Bangalore |
| 5 | Bhoomika JS | 1001 | 3000 | 1001 | Bangalore |
| 6 | Apoorva R | 1001 | 3000 | 1001 | Bangalore |

Figure 14 MySQL Query result for given problem statement Question 4

- v. Display the players and Team details , in which the *TeamNo* is same in both the players and *Team* (use natural join)

MySQL Commands

```

46 -- Question 5
47
48 select * from player natural join team;
49

```

Figure 15 MySQL Command for given problem statement Question 5

Result

| Team_No | Player_ID | Name | Score | Team_Name |
|-----------|-----------|-----------------|-----------|-----------|
| Filter... | Filter... | Filter... | Filter... | Filter... |
| 1001 | 1 | K Srikanth | 4000 | Bangalore |
| 1002 | 2 | Naveen Kumar GN | 3000 | Jaipur |
| 1003 | 3 | Sushanth | 3000 | Delhi |
| 1001 | 4 | P Supraja | 3000 | Bangalore |
| 1001 | 5 | Bhoomika JS | 3000 | Bangalore |
| 1001 | 6 | Apoorva R | 3000 | Bangalore |

Figure 16 MySQL Query result for given problem statement Question 5

- vi. Display the players and their team names, in which the *TeamNo* is same in both the players and *Team* (use left outer join)

MySQL Commands

```
50 -- Question 6
51
52 select play.name,t.Team_Name from player as play left join team as t on play.Team_No = t.Team_No;
53
```

Figure 17 MySQL Command for given problem statement Question 6

Result

| name | Team_Name |
|-----------------|---------------|
| abc Filter... | abc Filter... |
| K Srikanth | Bangalore |
| Naveen Kumar GN | Jaipur |
| Sushanth | Delhi |
| P Supraja | Bangalore |
| Bhoomika JS | Bangalore |
| Apoorva R | Bangalore |

Figure 18 MySQL Query result for given problem statement Question 6

- vii. Display the team names and the players involved, in which the *TeamNo* is same in both the players and *Team* (use right outer join)

MySQL Commands

```
54 -- Question 7
55
56 select play.name,t.Team_Name from player as play right join team as t on play.Team_No = t.Team_No;
57
```

Figure 19 MySQL Command for given problem statement Question 7

Result

| name | Team_Name |
|-----------------|---------------|
| abc Filter... | abc Filter... |
| K Srikanth | Bangalore |
| P Supraja | Bangalore |
| Bhoomika JS | Bangalore |
| Apoorva R | Bangalore |
| Naveen Kumar GN | Jaipur |
| Sushanth | Delhi |

Figure 20 MySQL Query result for given problem statement Question 7

Question C

Java Code

Find_Button (Home.java)

```

151 private void Find_ButtonActionPerformed(java.awt.event.ActionEvent evt) {GEN-FIRST:event_Find_ButtonActionPerformed
152     String Name;
153     int c;
154     Name = Input_Field.getText();
155     try {
156         Class.forName("com.mysql.cj.jdbc.Driver");
157         con = DriverManager.getConnection("jdbc:mysql://localhost:3306/lab_7","root", "Sri123");
158         pst = con.prepareStatement("select Team_No,Name from player as p1 where p1.Team_No = (select Team_No from player as play where play.name = ? );");
159         pst.setString(1, Name);
160         Input_Field.setText("");
161         rs = pst.executeQuery();
162         ResultSetMetaData rsd = rs.getMetaData();
163         c = rsd.getColumnCount();
164         DefaultTableModel dft = (DefaultTableModel) Result_Table.getModel();
165         dft.setRowCount(0);
166         while (rs.next()) {
167             Vector v2 = new Vector(); Vector is a raw type. References to generic type Vector<E> should be parameterized
168             for (int i = 1; i < c; i++) {
169                 v2.add(rs.getString("Team_No")); Type safety: The method add(Object) belongs to the raw type Vector. References to generic type Vector<E> should be pa
170                 v2.add(rs.getString("Name")); Type safety: The method add(Object) belongs to the raw type Vector. References to generic type Vector<E> should be param
171             }
172             dft.addRow(v2);
173         }
174     } catch (ClassNotFoundException ex) {
175         Logger.getLogger(HomePage.class.getName()).log(Level.SEVERE, null, ex);
176     } catch (SQLException ex) {
177         Logger.getLogger(HomePage.class.getName()).log(Level.SEVERE, null, ex);
178     }
179 }
180 } //GEN-LAST:event_Find_ButtonActionPerformed
181

```

Figure 21 Java Program for Find Button (Action)

Scenario 1 (Question 1)

UI/JFrame

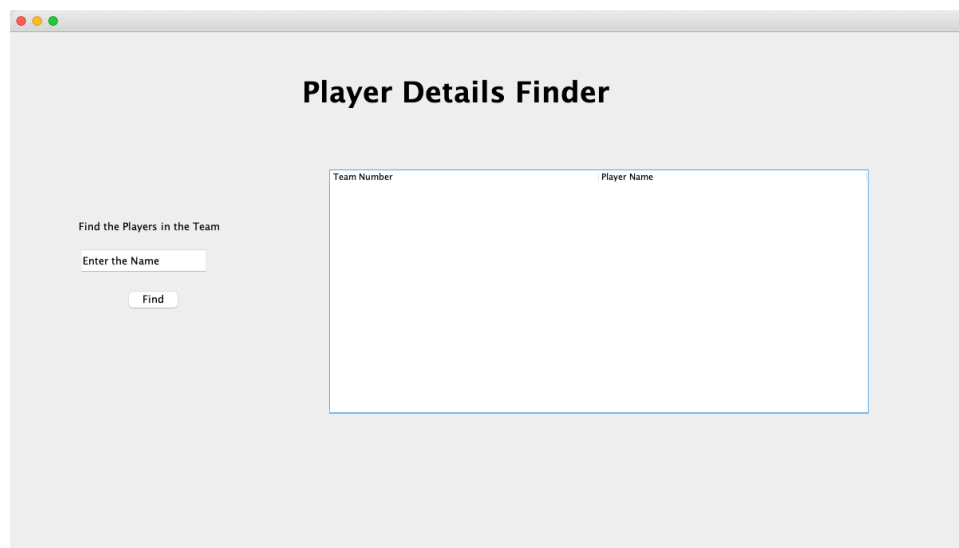
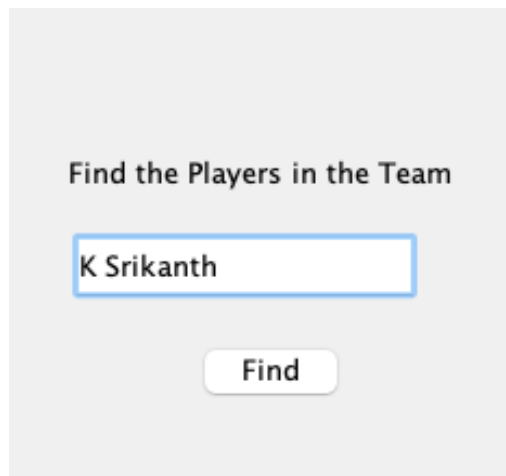


Figure 22 Swing UI using Java (Home Page)

This is the basic UI layout (Home Page) where we can see that it has **one text field** where the user can enter the player **name** and there is one button which is **find** and when you click on it, it will execute the query and will display it on the table next to it.

Now let's Enter the data into the Text Fields Our **Player Name is "K Srikanth"** and when we hit find button it will execute a SQL Query which will display all the player names in which **"K Srikanth" is present.**



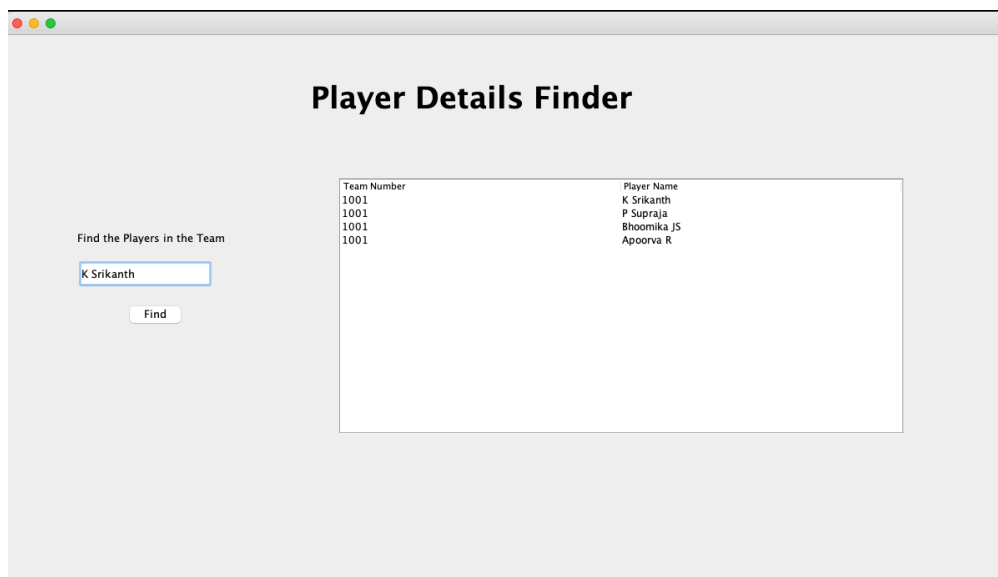
Find the Players in the Team

K Srikanth

Find

Figure 23 Name Field To execute the following query (Figure 7)

As we can see from Image 24 that we can see all the players which belong **"K Srikanth"** along with their Team Number



Player Details Finder

Find the Players in the Team

K Srikanth

Find

| Team Number | Player Name |
|-------------|-------------|
| 1001 | K Srikanth |
| 1001 | P Supraja |
| 1001 | Bhoomika JS |
| 1001 | Apoorva R |

Figure 24 Updated Table when the Query is executed

6. Conclusions

Nested queries are basically a query is written inside a query as the result of inner query is used in execution of outer query.

Joins

1. Natural Join

Natural Join joins two tables based on same attribute name and datatypes. The resulting table will contain all the attributes of both the table but keep only one copy of each common column.

2. Inner Join

Inner Join joins two table on the basis of the column which is explicitly specified in the **ON** clause. The resulting table will contain all the attributes from both the tables including common column also.

3. Left Join

This join returns all the rows of the table on the left side of the join and matching rows for the table on the right side of join. The rows for which there is no matching row on right side, the result-set will contain null. **LEFT JOIN** is also known as **LEFT OUTER JOIN**

4. Right Join

RIGHT JOIN is similar to **LEFT JOIN**. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of join. The rows for which there is no matching row on left side, the result-set will contain null. **RIGHT JOIN** is also known as **RIGHT OUTER**

7. Comments

Learning happened

Joins, nested queries and their application in solving problems.