# 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.

1. 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

1. 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
2. Questions
   1. 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)

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

1. **Presentation of Results**

**Question A**

**MySQL Commands**

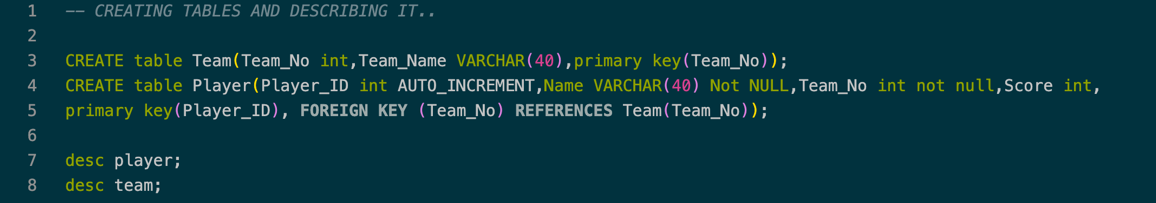
****

Figure 1 MySQL Command to Create TEAM and PLAYER TABLE

**Result**

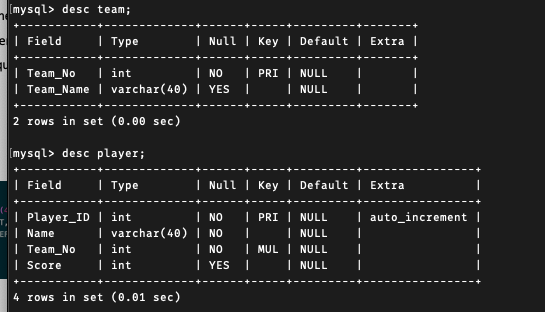
****

Figure 2 Meta-Data of PLAYER and TEAM Table

**Inserting data into tables**

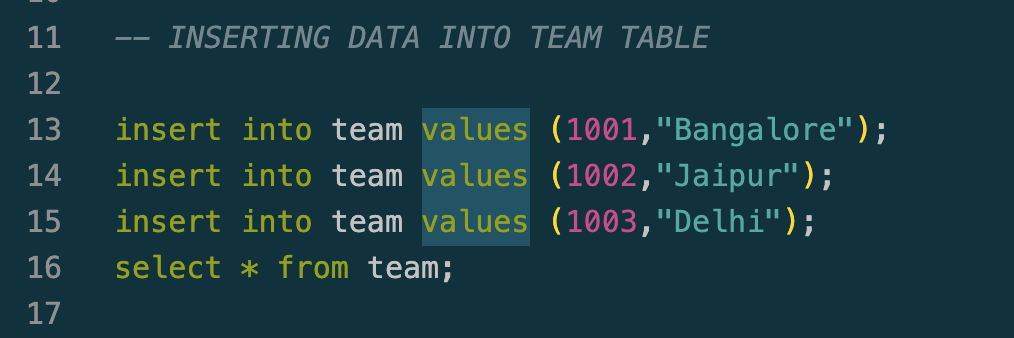
****

Figure 3 Insert Operation on TEAM Table

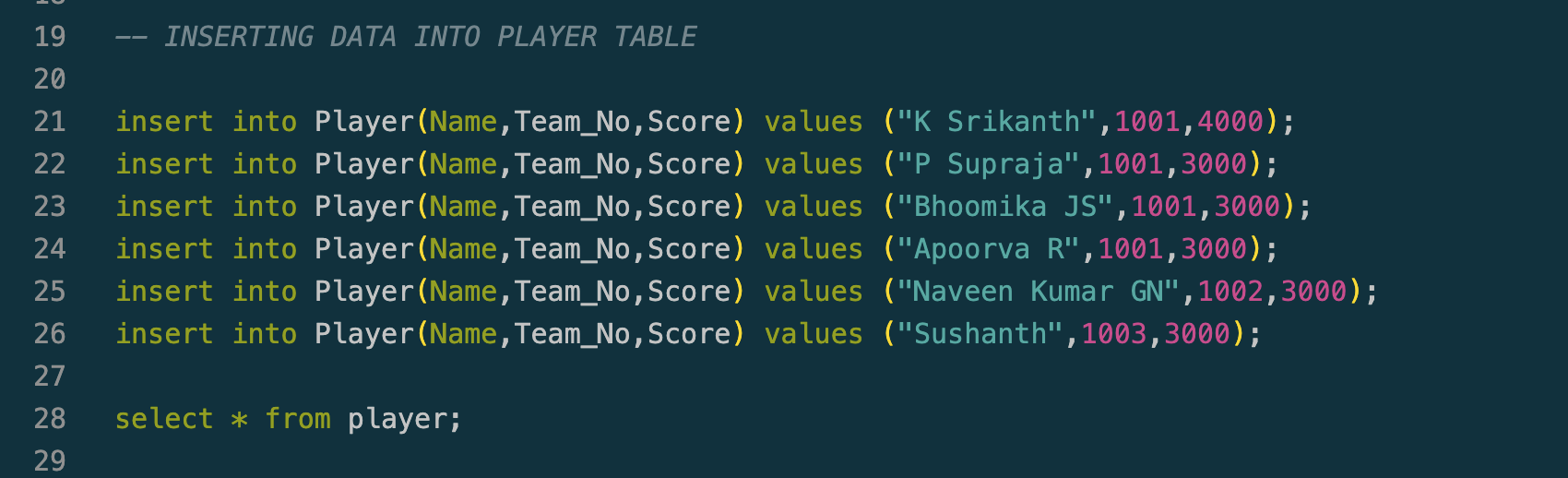
****

Figure 4 Insert Operation on PLAYER Table

****

Figure 5 Select Operation on PLAYER Table

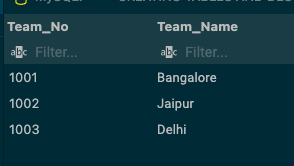
****

Figure 6 Select Operation on TEAM Table

**Question B**

* + 1. Find the names of all the players who are in the same Team of ‘Smith’ (use nested query)

**MySQL Commands**

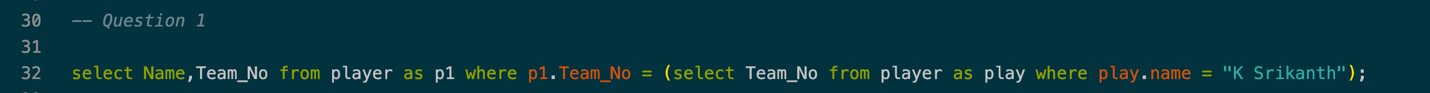
****

Figure 7 MySQL Command for given problem statement Question 1

**Result**

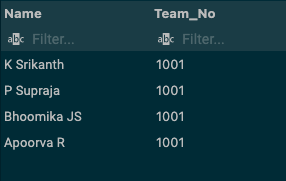
****

Figure 8 MySQL Query result for given problem statement Question 1

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

**MySQL Commands**

****

Figure 9 MySQL Command for given problem statement Question 2

**Result**

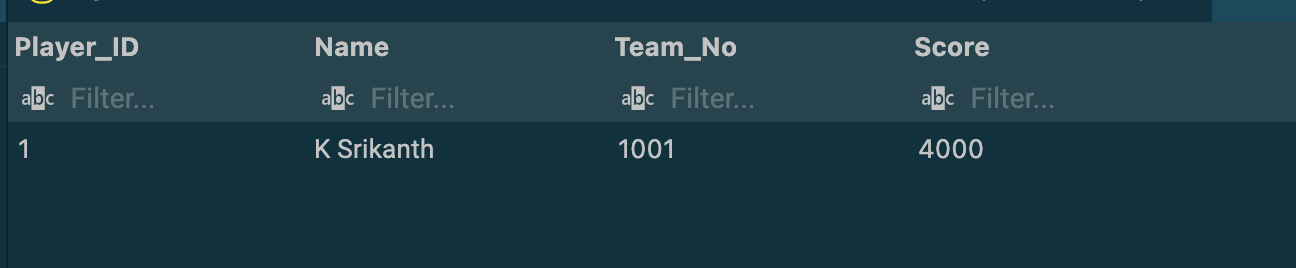
****

Figure 10 MySQL Query result for given problem statement Question 2

* + 1. Display the players and Team details , in which the *TeamNo* is same in both the playersand *Team* (without join)

**MySQL Commands**

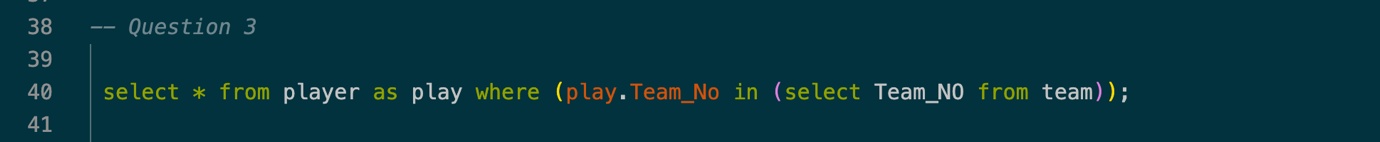
****

Figure 11 MySQL Command for given problem statement Question 3

**Result**

****

Figure 12 MySQL Query result for given problem statement Question 3

* + 1. Display the players and Team details , in which the *TeamNo* is same in both the playersand *Team*

**MySQL Commands**

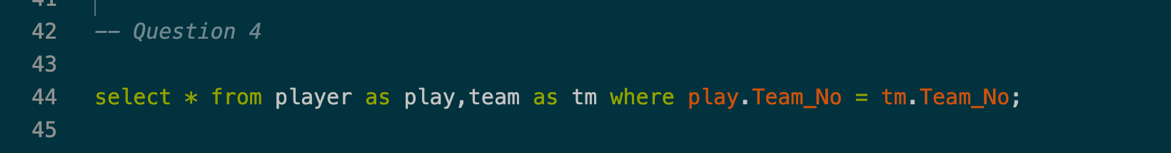
****

Figure 13 MySQL Command for given problem statement Question 4

**Result**

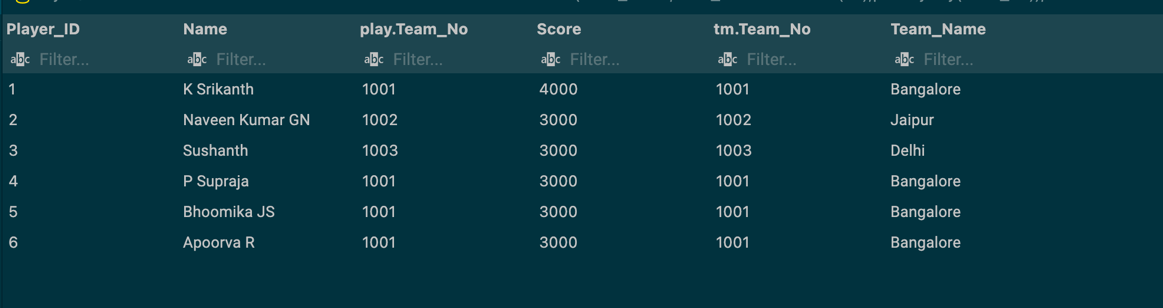
****

Figure 14 MySQL Query result for given problem statement Question 4

* + 1. Display the players and Team details , in which the *TeamNo* is same in both the playersand *Team* (use natural join)

**MySQL Commands**

****

Figure 15 MySQL Command for given problem statement Question 5

**Result**

****

Figure 16 MySQL Query result for given problem statement Question 5

* + 1. Display the players and their team names, in which the *TeamNo* is same in both the playersand *Team* (use left outer join)

**MySQL Commands**

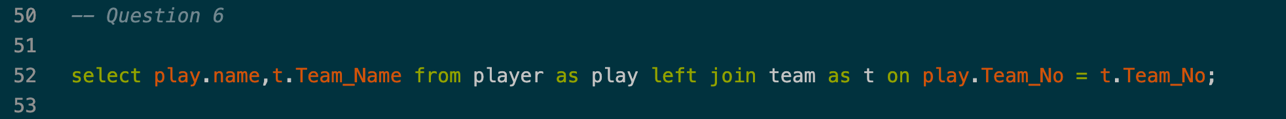
****

Figure 17 MySQL Command for given problem statement Question 6

**Result**

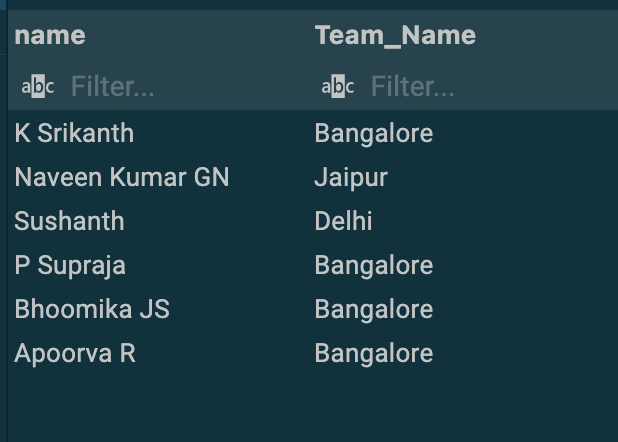
****

Figure 18 MySQL Query result for given problem statement Question 6

* + 1. Display the team names and the players involved, in which the *TeamNo* is same in both the playersand *Team* (use right outer join)

**MySQL Commands**

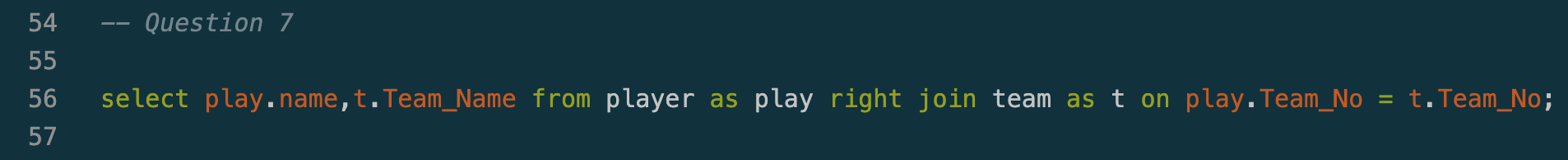
****

Figure 19 MySQL Command for given problem statement Question 7

**Result**

****

Figure 20 MySQL Query result for given problem statement Question 7

**Question C**

**Java Code**

**Find\_Button (Home.java)**

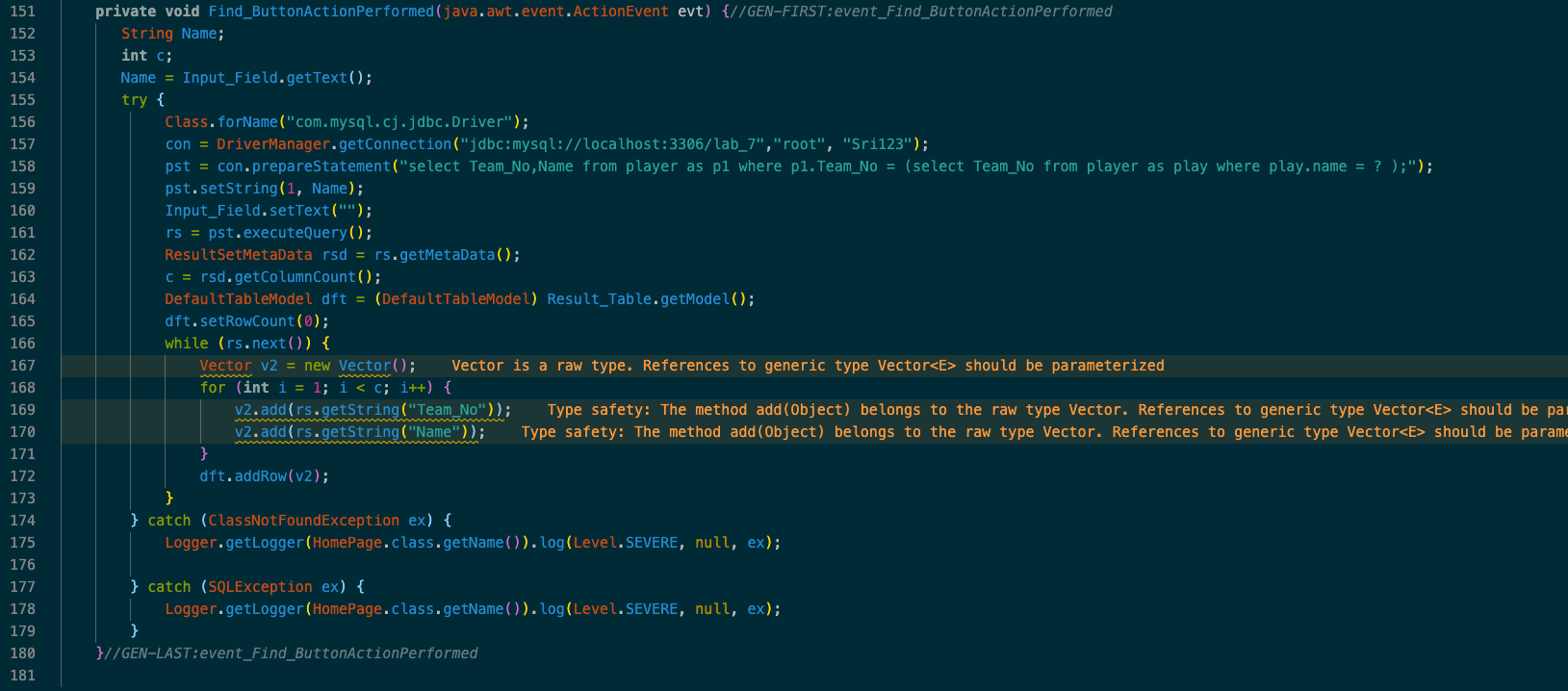
****

Figure 21 Java Program for Find Button (Action)

**Scenario 1 ( Question 1 )**

**UI/JFrame**

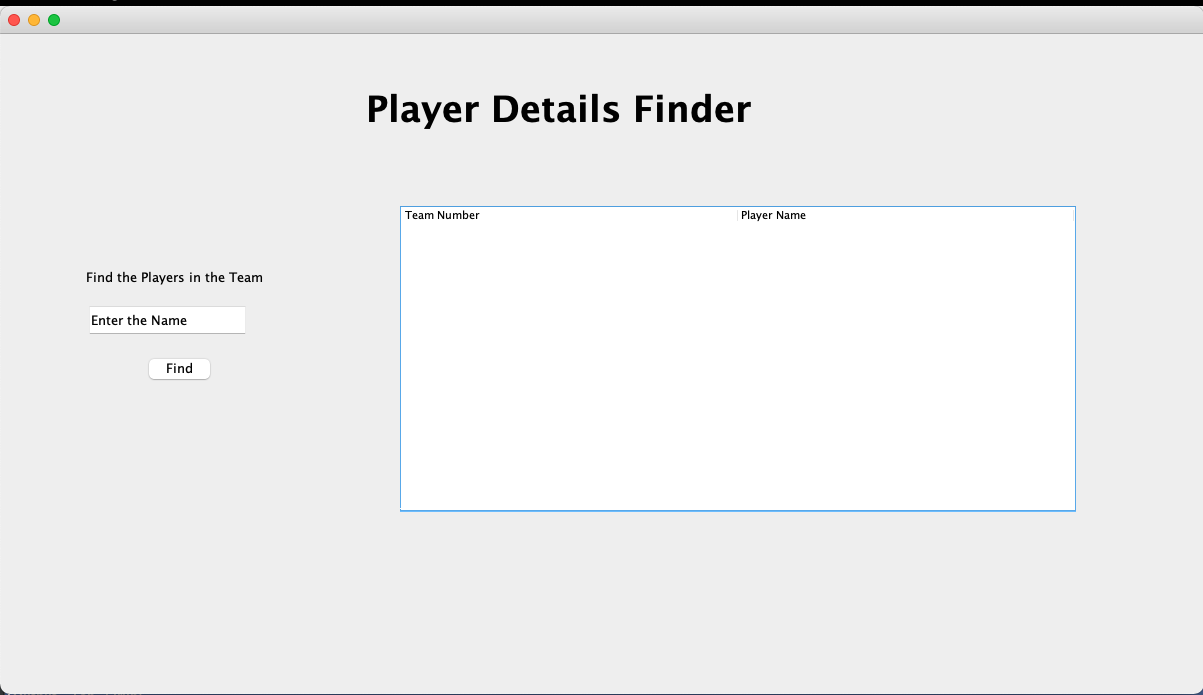
****

Figure 22 Swing UI using Java (Home Page)

This the basic UI layout (Home Page) where we can see that it have **one text field** where the user can enters the player n**ame** and there are one button which is **find** and when you click on 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.**

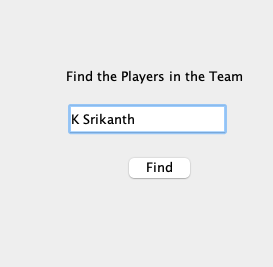
****

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

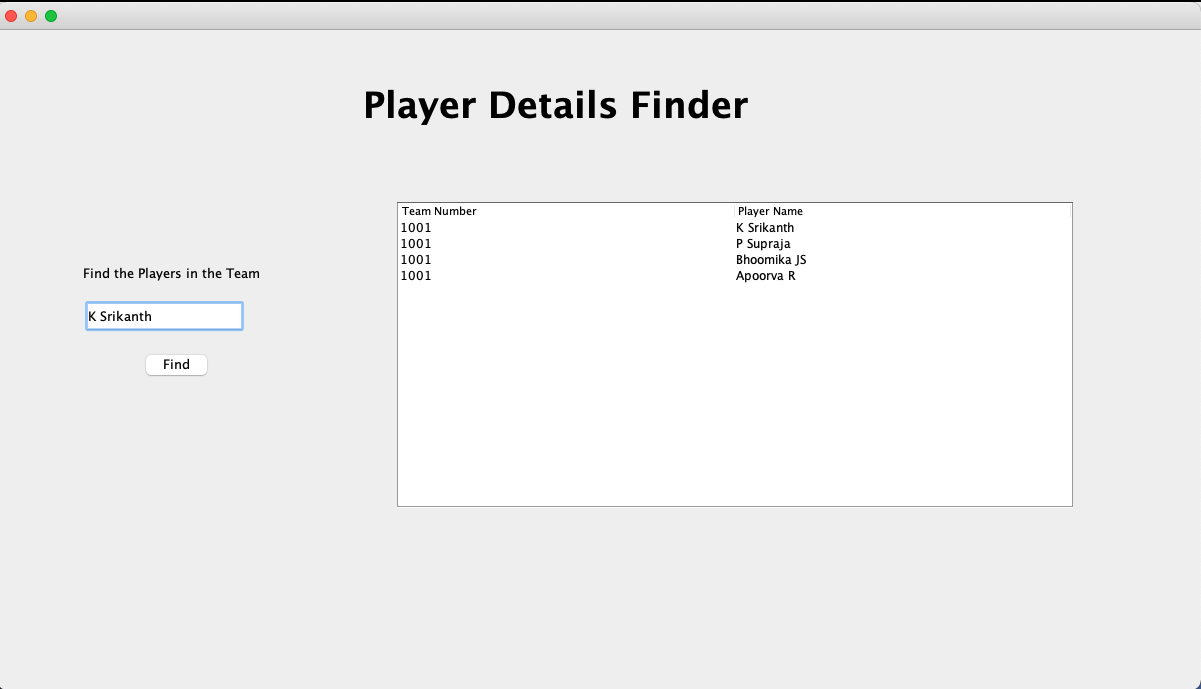
****

Figure 24 Updated Table when the Query is executed

1. **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**

1. **Comments**

**Learning happened**

Joins, nested queries and their application in solving problems.