Dhirubhai Ambani
Institute of Information and Communication Technology

Basketball Training Management System

Team: S6_T1

IT214 Database Management System, Autumn' 2021

Course Instructor: Prof. Minal Bhise

Mentor TA: Kavan Sir

Group Members:

Nilay Shah **ID-201901026**

Kathan Sanghavi **ID-201901053**

Ayush Prajapati **ID-201901099**

Darshan Sohaliya **ID- 201901219**

Index

Section 1

1) Final Version of SRS

[A] Complete Description of the Case Study/Problem Domain

1. Introduction

1.1 Purpose

Our purpose is to provide various functionalities for the **Basketball Training Management System**. This document provides constraints, which system will follow. The final product will have features/functionalities mentioned in this document.

1.2 Intended Audience and Reading Suggestions

This SRS is intended for all types of the reader such as players, developers, users, staff, marketing staff, and product managers. This SRS contains the complete description of the system and user interfaces. It also contains user characteristics and authorization for player, head coach and trainer login. It also contains information about player and coaches, and also have a screen for event-related information.

- The best way to read this document is to start from the introduction section. Then developers can read about the system and user interfaces.

- Players can check out user characteristics and their constraints and the player and trainer's information screen. Then they can also check about training sessions.

1.3 Product Scope

The software "**BASKETBALL TRAINING MANAGEMENT SYSTEM**" will be an application that will be used for managing basketball databases. The application will manage the information about the players, the matches and the events that are to be held, coaches and the board that will plan everything. It will also store data on the physical status of all the players. This system will provide better data integrity, faster data access, minimum data inconsistency and updating data of all players and all related members. The main objective of this project is to provide a more efficient and productive way of scheduling and monitoring Basketball training sessions.

This software is also convenient for them to announce the flow of events and other matters involved in the training sessions and practice matches held because they have the records which are essential for hosting a Basketball and they can still keep a control out of the unexpected incidents that may occur during the event. This kind of sports training management system is also very attractive in commercial sectors.

1.4 Description

The database is about “Basketball Training Management System”. This system will enable us to identify, clarify and organize relationships between various entities. This system describes how the administrator manages the player, trainer, Head Coach and other different entities. Several different attributes are interrelated to each other with some relationships. The efficiency of the basketball tournament scheduling, training session, provision of schedules was the main purpose of creating this database system. This system will save schedules of all the activities with less consumption of time and effort.

For Basketball Training Management System, we need to constantly monitor the level of skills of players. This must be the focus of our application. To accurately monitor and provide better feedbacks to players, we should divide training session into various types. Player should be graded by trainer in each session, so that player can get continuous report on improvement in his level in various skills. As a training management center, academy should also focus on player’s performance in real matches. Thus, our application also provides functionality to rate player based on his performance in practice match as match power skill. Trainers and Head Coach are given right to give grades to player based on players’ performance in each skill area. To analyze player’s performance easily, players, trainers and Head Coach are provided with graphs on players’ progress, so that they can easily track players’ progress in various skills. Head Coach is also provided with graph on overall skill level of academy so that he can examine the progress of academy as a whole.

Players are also provided with ranks in various skills based on average of grades given to them. So that players’ can get comparatively better. And players will always be motivated to continuously improve their skills to get better ranks.

Match Power skill is the most important skill that we have decided to include skill set of player. As in the end performance in the match – contribution – impact made by the player, are the things which will matter. Thus trainer will monitor players’ performance in the match, and then he will give grades to the player based on his performance in the match. Other skills will be Dribbling, Shooting, Defence, Rebounding, Passing and Fitness. Players will get overall grade based on grades in each skills. Thus players will also get overall rank, based on overall grades.

So that courts don’t become overcrowded. As if the number of players on a court present at one time, exceeds the certain threshold then it won’t be feasible to give proper space to each player.

Academy should provide personal attention to each and every player. Personal guidance to players’ is very crucial for players’ progress. Thus we should limit the number of players, one trainer can teach in one session at a time.

Application will also provide a separate screen for matches’ related information. List of teams and their points will be visible to everyone on that screen. We should include screen related practice matches as, matches are fun to play and players will get continuous motivation to improve their skills and get win in the matches for their team.

Other details are as follows: Players will be provided with grade cards for every month. There are different salary structures and fee structures. Player is assigned to one fee structure and trainer is assigned to one salary structure. Fee structure and salary structure are assigned based on experience of player or trainer.

Below is the basic description of role of each type of user in database.

Operating Environment: Web

Users:

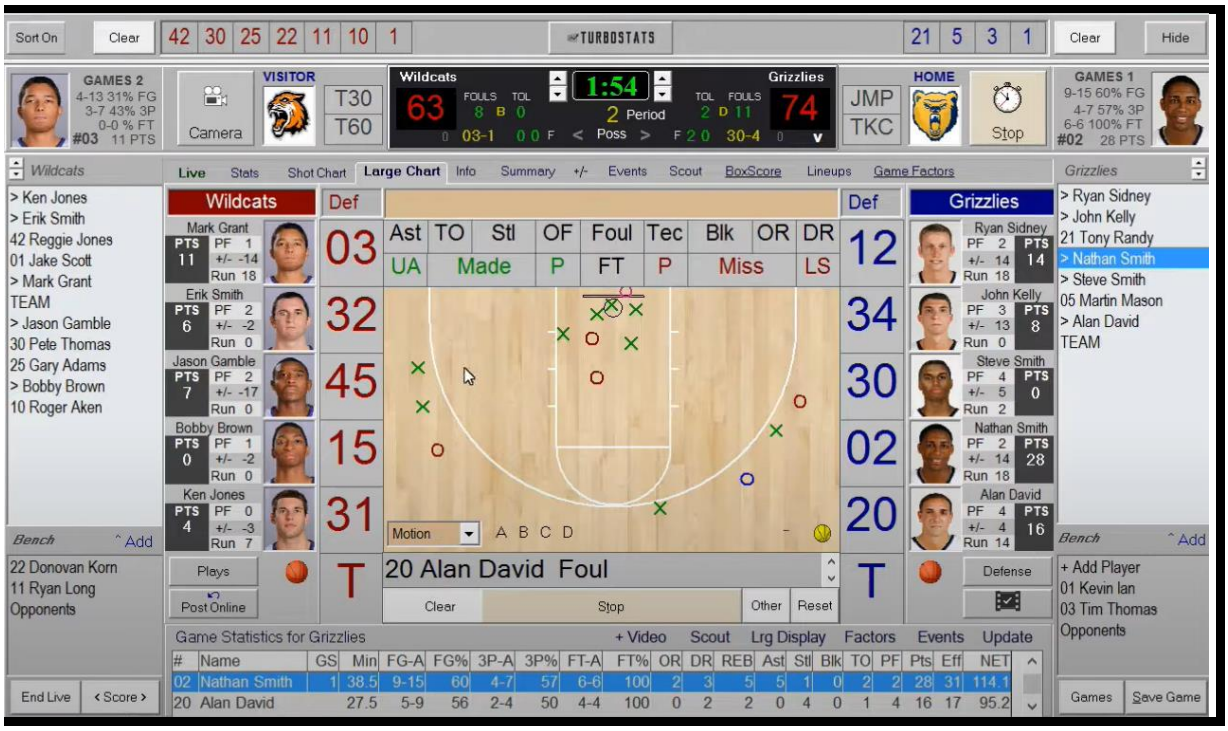
1) Player:
    - Players will be provided with a dashboard having a training schedule with trainers assigned to them and training session type for those sessions and a progress chart.
    - The Dashboard will also have details about Payment Gateway, Current Application Status.
    - Players won't have additional access other than his profile.
2) Trainer:
    - The Trainer will have to login through the login screen by entering id and password.
    - The Trainer will be provided with a dashboard having a training schedule with Players assigned for the session.
    - The Trainer will have limited read access to the contact information of players and other trainers.
3) Head Coach
    - After login, the head coach has edit access to everyone's schedule, can change fees - salary structure, approve/disapprove the application of players and trainers.
4) IT Administrator:
    - As the admin of the system, he has full edit access and authority to grant access to others.

[B] Document the Requirements Collection/ Fact-Finding Phase

[1] Background Reading:

Basketball is a traditional sport and platform to enhance the feeling and mutual exchange. It is a very popular sport and at the same time, it is an international sports event, either in NBA or CBA. For this sport, we need more professional basketball players, which bring the audience a wonderful visual effect but also put forward higher requirements to the basketball players' training institutions. The Beijing 2008 Olympic Games has also promoted the sport. Therefore, we need to design such a system that gives more efficiency and consistency. The uses of training resources are very low due to the traditional Basketball management system. To improve the performance of the traditional basketball training system, we introduce this database system. Manual ways of doing activities like transaction evaluation of all the training session evaluation were unable to provide such a schedule of a training session with trainees, schedule with trainers and venues in a small period. By the traditional way, it is also very difficult to concurrently access and change the data manually and revert the decision among the data. Aside from that, it is not secured, tracing these transactions was also hard for the administrators because they lack back-ups. Informing the participants involved in the tournament will also consume a lot of time and effort for the administrators of this system. To improve on such things and provide the best user experience, this system will be very useful and efficient. In modern times, we have fully-featured Statistical Analysis Software designed that allows players to calculate statistics related to their skills and efficiency in a match and gives the summarized result of their overall skill. We will add some more functionality and additional features in the direction of concurrency and security and atomicity of updates.

Case Study 1: TURBOSTATS

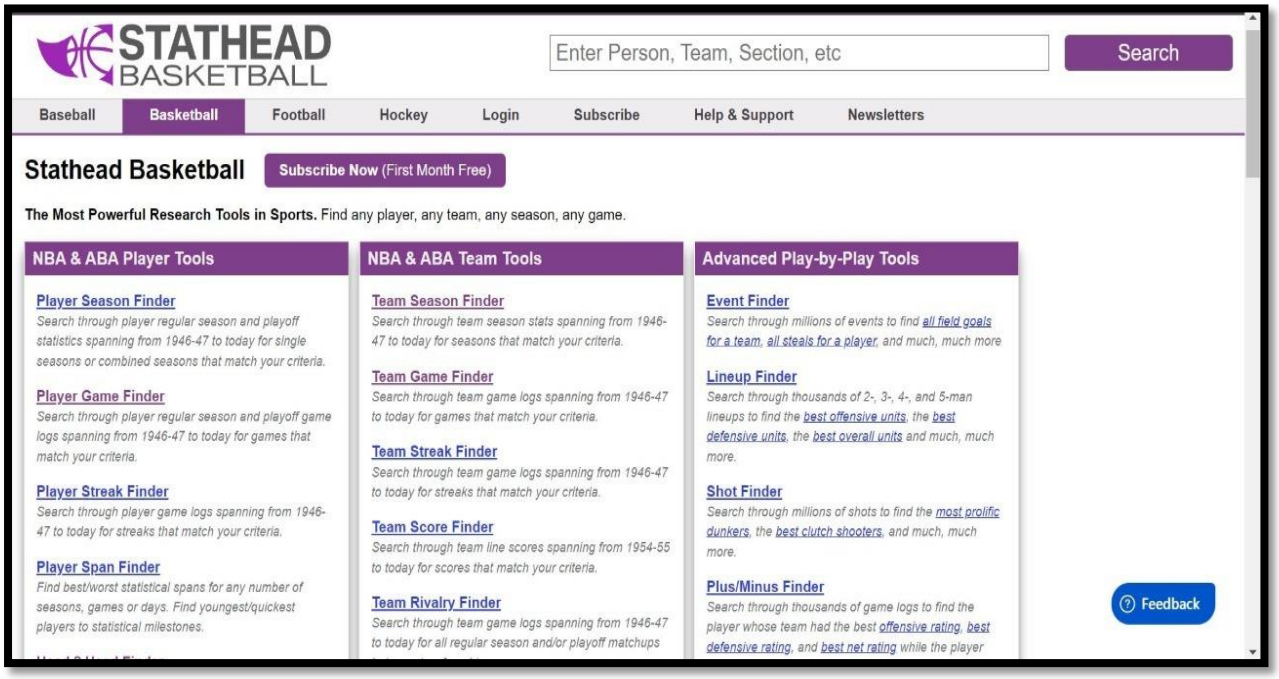


Source:

http://www.turbostats.com/basket.htm#ls-link

- Turbo Stats allows users to calculate statistics related to games and scores.
- The software has the capability to track over 300 statistics and other metrics related to the players performing.
- The software has the capability to perform statistical analysis to make rapid decisions that lead to victory situations.
- The software allows one to view live scores and performance.
- Users can perform analysis to improve their team performance and identify flaws that are hindering and obstructing performance.
- The software allows users to make animations. It can animate players and their actions.
- It enables us to ensure that every team member is available and be on time.
- The software is capable of working offline as well as online.
- It develops charts so that every player can measure their performance and try to make the performance better than previous ones.
- Turbo Stats is more oriented towards match analysis and doesn't provide many functionalities for training management.

Case Study 2: STATHEAD



- STATHEAD BASKETBALL allows users to know the information of players such as Player season, game, streak, span, quarter.
- It can show Team's information such as Team season, game, streak, score, and rivalry.
- It has the feature of a player comparison finder which can compare 2 players' performance.
- STATHEAD BASKETBALL can show players age, position, Team, Points, Field goals, Free throw attempts and a lot more.
- It has a lot of filters related to basketball information which can give a specific result.
- It also contains head's information.
- It has Data Coverage Summary which contains all information about all team's information, points, seasons etc.
- STATEHEAD BASKETBALL provides good features regarding skill and performance analysis but lacks some features regarding training management like session scheduling.

References:

1. *http://article.nadiapub.com/IJDTA/vol9_no12/27.pdf*
2. *https://itsourcecode.com/fyp/automated-basketball-scheduling-and-monitoring-system-chapter-1/*
3. *https://studentprojectguide.com/vb-net/sports-management-system/*
4. *https://itsourcecode.com/fyp/automated-basketball-scheduling-and-monitoring-system-methodology/*
5. *http://www.turbostats.com/basket.htm?gclid=CjwKCAjwqeWKBhBFEiwABo_XBgDMhyY_rvHcoJWlbTG60g9aD7wvTJXtUHQpeBjqtM_653jKrtYv6RoC0qoQAvD_BwE*
6. *https://www.predictiveanalyticstoday.com/turbostats/*
7. Player Game Finder | Stathead.com
8. https://stathead.com/basketball/

Requirements gathered from Background Readings:

- Transaction Security and full access only to specific users like administrators.
- Data integrity.
- Referential Integrity.
- Concurrent Access needed to update the related data changed by administrator.
- Concurrent Update.
- Past match statistics for the trainee reference to make their game much better and competitive.
- Functionalities like Player Statistics and Player Comparisons and Coach Information.
- Easy to use User Interface with ample amount of information.

[2] Interviews:

Mock Interview Summary

System: Navrang Basketball Academy

Interviewee: **Name:** Nirav Shah

Designation: Head Coach (Navrang Basketball Academy)

Date: 4/10/2021 Time: 17:30

Duration: 1 Hour

Place: Head Coach's Office, Navrang Basketball Academy

Purpose of Interview:

1. Currently, no custom application for training management.
2. They are using excel sheets to store players' and trainers' information. Currently, they aren't sharing these sheets with players for security reasons. Thus, players don't have even read access currently.
3. Currently, few trainers started accessing players' details and schedules, then started contacting students for personal coaching tuitions without the permission of the head coach. As trainers can get all information about all players from excel sheets, this creates security risks and violates players' privacy.
4. Sometimes basketball courts become overcrowded, as currently, they have no method to count the total number of players present in the court, at any time.
5. Get information regarding session timings and session duration.
6. Schedule an interview with some players and trainers.

Requirements gathered from Interview:

- Players should not be able to see other players' details.
- The system should not allow changes that create conflicts in schedules.
- Custom Restriction, limits the amount of minimum and maximum training time for a player.
- Head Coach should be able to make changes in any player's or trainer's schedule.

- The application should automatically change player status to inactive if fees are not paid on time by the player.

Mock Interview Summary

System: Navrang Basketball Academy

Interviewee: **Name:** Mayur Jani

Designation: Administrator (Navrang Basketball Academy)

Date: 4/10/2021 Time: 18:35

Duration: 1 Hour

Place: Administrator Office, Navrang Basketball Academy

Purpose of Interview:

1. They have to check manually for conflict in schedules, currently. This process takes a lot of time.
2. Since players can't see available slots, they make random requests, which are not feasible.
3. Web-Based applications can provide an easy way to check restrictions on the minimum amount of training and the maximum amount of training.
4. As the number of players increases, an old method used by them gets harder to manage.

Requirements gathered from Interview:

- Custom Restriction limits the amount of minimum and maximum training time for a player.
- Head Coach should be able to make changes in any player's or trainer's schedule.
- There should be restrictions on the number of players in one court. Changes that violate this, should not be allowed.
- Available slots with trainer names can be seen.
- The database should also have all information about the player's default position in court, height, age, etc. And also the player's rating in each skill.
- The application should also generate attendance reports of players.

Mock Interview Summary

System: Navrang Basketball Academy

Interviewee: **Name:** Satish Sharma

Designation: Trainer (Navrang Basketball Academy)

Date: 4/10/2021 Time: 20:00

Duration: 30 minutes

Place: Court 3, Navrang Basketball Academy

Purpose of Interview:

1. Currently, no easy mechanism to track players' progress. There should be graphs and statistics regarding players' progress in new web applications.
2. There should be an attendance report for players in the new web application. Currently, there isn't any.

Requirements gathered from Interviews System:

System: Navrang Basketball Academy

Participants: Satish Sharma (Trainer at Navrang Basketball Academy)

Requirements gathered from Interview:

- Trainers should be able to see players' strengths and weaknesses.
- Trainers or players can request the head coach to make changes in their schedules, but they can't change it on their own.

Mock Interview Summary

System: Navrang Basketball Academy

Interviewee: **Name:** Sahil Patel

Designation: Player (Navrang Basketball Academy)

Date: 4/10/2021 Time: 20:35

Duration: 30 minutes

Place: Court 1, Navrang Basketball Academy

Purpose of Interview:

1. Get players' and trainers' perspective and their requirements from the new web application.
2. Discuss rescheduling the session request.

Requirements gathered from Interview:

- The training session can be of different types, such as Dribbling, Shooting, Defence, Rebounding, Passing, Fitness Session, and Practise Match Session.
- The trainer will rate the player's performance on a scale of 1 to 10, for each session.
- The head coach will analyze the progress of the academy and individual players. The application should generate reports and graphs for easy analysis.

[3] Questionnaires:

i. **Do you have any experience in Basketball?**
This question allows us to understand how many players are a newbie and how many are experienced. Thus we can make functionalities of the app according to that.

ii. **Which position will you prefer to play in basketball?**
This question will give us the exact position that the player wants to be in so that we can get some information regarding the interest of the players.

iii. **Have you heard about the Basketball training Management system?**
To know about how many people are aware and know about training management system.

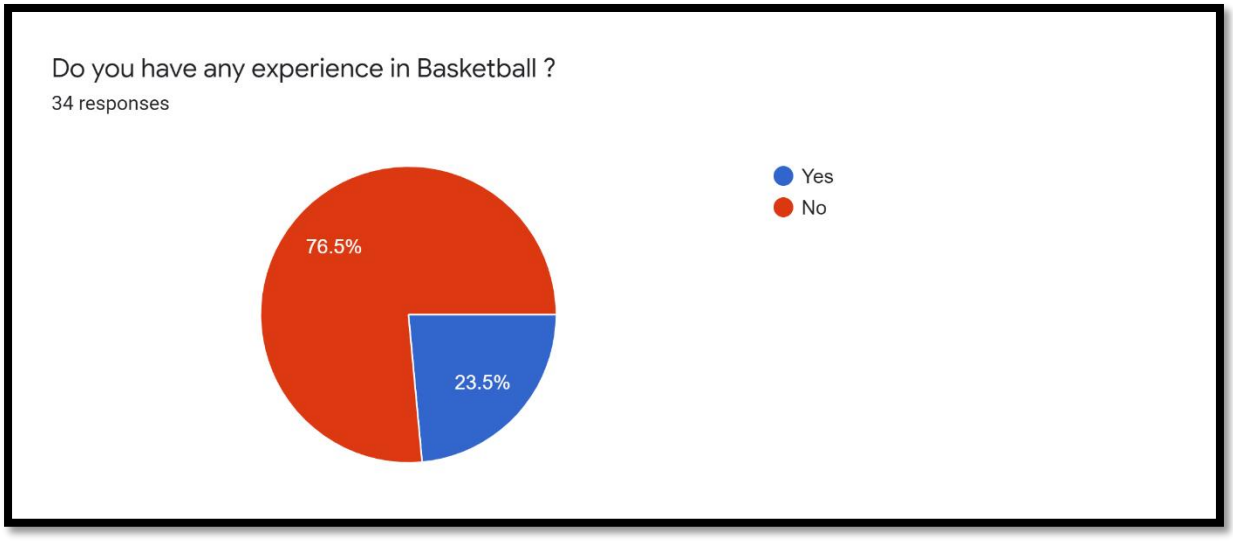
iv. **What is the name of the System you have heard about?**
This will give us some other system examples of a training management system.

v. **What is the important constraint that you think might be the most important for this database design?**
To know which functionalities are more important and reliable for the academy.

vi. **What is the most suitable benefit for you by using this kind of training management system?**
This question will give responses about what is the prioritized functionality of users.

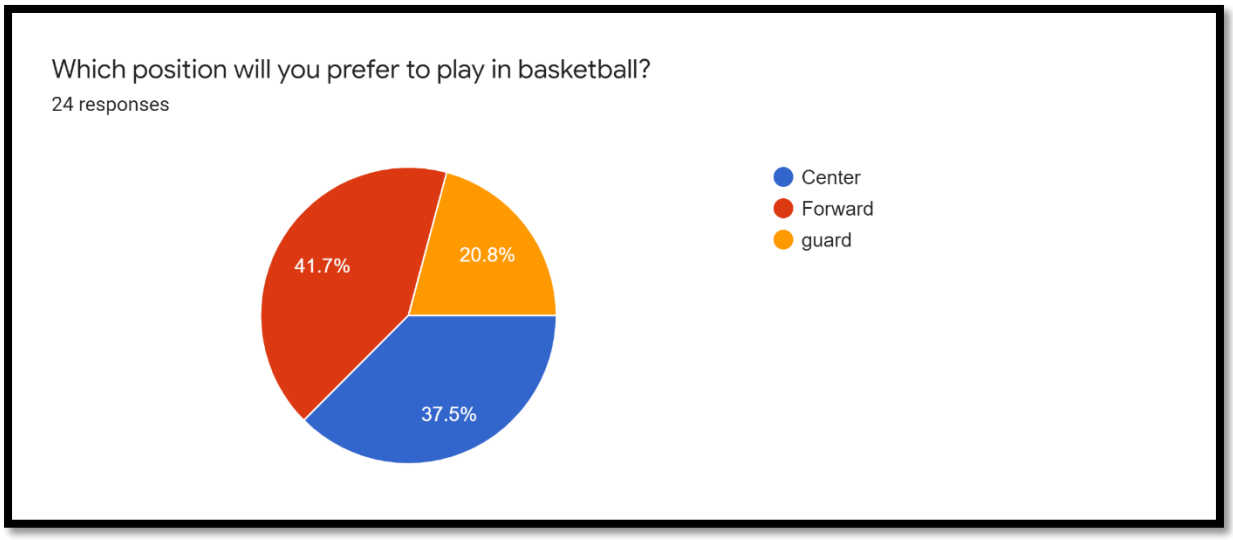
Responses and Observations:

From the below chart we can easily say that from our total users 76.5% of users have no experience. They are newbies in basketball. So we can give them according to the trainer and design functionalities based on the fact that they don't have much prior knowledge regarding Basketball. Rest 23.5% of users already have knowledge about Basketball, so we will give them trainers who can give them training from the intermediate level.

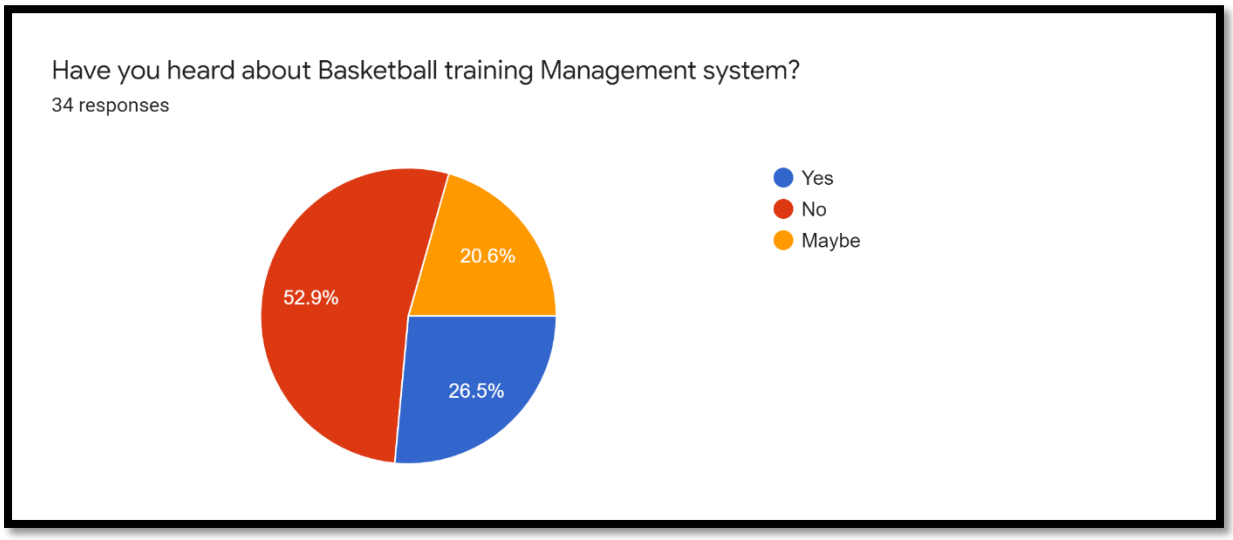


The responses to these questions will give us the idea that which player should get which coach according to their interested position to play basketball. Here 41.7% of players want to play at forward position, 20.8% want

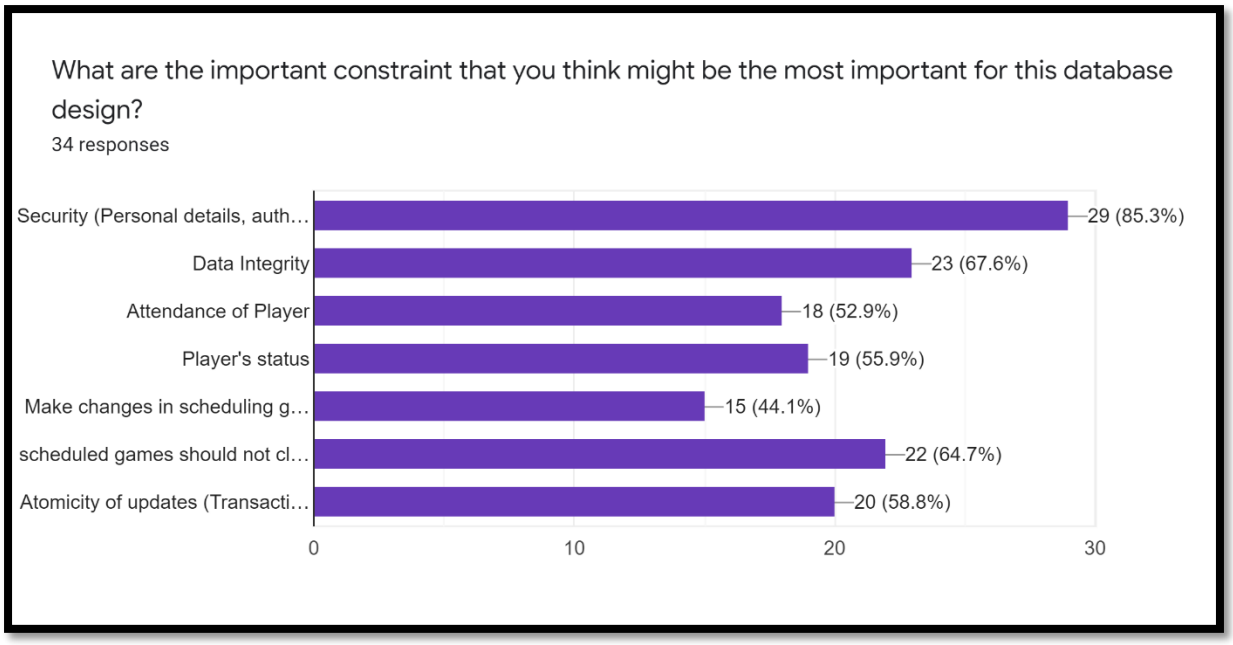
guard as position to play basketball and 37.5% of players want center position. We can easily assign trainers according to this information.



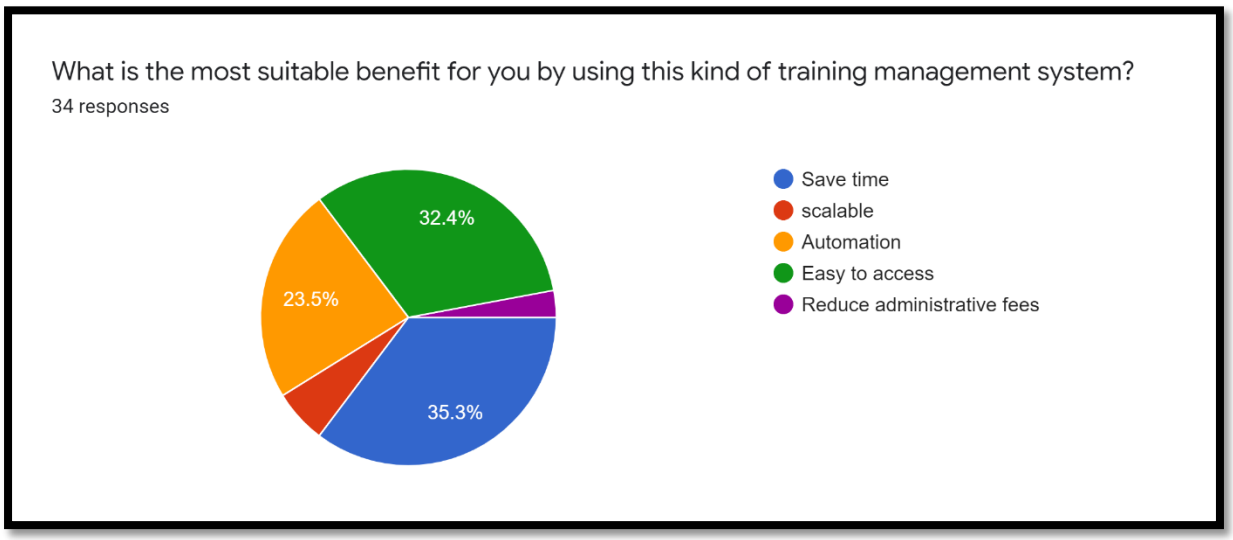
Below chart will give us the information about how many players are already aware of training management system and how many are not aware of this. In further to this question, we also asked a question which shows that how do you know about training management system and the response is only from web.



This chart shows that security is the most important concern for the users. 85.3% of the players want security as a priority. Data integrity is also selected by 67.6% of players. These 2 functionalities should be focused on while making a database.



From the below chart, the most suitable benefit for the user is time-saving (35.3%). It gives us clue that the database should be very accurate and fast. The second most benefit is easy to access (32.4%). The database should be well designed so all users can access their requirements as easily as possible.



Requirements gathered from Questionaries:

- Out of our total users, 76.5% of users have no experience. They are newbies in basketball. So we can give them according to the trainer and design functionalities based on the fact that they don't have much prior knowledge regarding Basketball.
- Security is the most important concern for the users.
- The most suitable benefit of users of Basketball Training Management Application time saving (35.3%).
- By taking the survey, we come to know that Web-based application is very popular among the people.

[4] Observations:

1. Security is not that good in the old method.
2. Due to the lack of security anyone can access other person's profiles.
3. The authority to access other players' profiles must be only limited to the trainer and head coach.
4. The authority to schedule all the matches/training sessions should be only to the Head coach.
5. The match/training schedules are not inclined with each other. There are so many clashes between matches.
6. There is not that good health facility for injured players.

[C] Fact-Finding Chart

Objective	Technique	Subjects	Time Commitment
To get the idea about software available in the market	Background Reading	○ https://stathead.com/basketball/ ○ http://www.turbostats.com/basket.htm?gclid=CjwKCAjwqeWKBhBFEiwABo_XBgDMhyY_rvHcoJWlbTG60g9aD7wvTJXtUHQpeBjqtM_653jKrtYv6RoC0qoQAvD_BwE	2 days
To get a basic idea about the requirements of a basketball training centre from an application	Interview	Head Coach at Navrang Basketball Academy	1 hour
To get the idea of what difficulties players are facing with the old method of managing data	Interview	Basketball Player	3 x 30 mins each
Get an idea about trainers perspectives, regarding rescheduling training sessions and attendance reports or other such functionalities for tracking the progress of a player	Interview	Basketball Trainer at Navrang Basketball Academy	2 x 30 mins each
Get an idea about data security and the amount of players information that needs to be stored	Interview	Head Coach at Navrang Basketball Academy	1 hour
Discussion regarding the old method used by the academy to manage data	Interview	Administrator at Navrang Basketball Academy	1.5 hour

and flows in that method, Improvements wanted in a new system. Discussion regarding hardware/software costs			
General ideas and expectations of players/people for Basketball Training Management Application	Questionnaire – Google Form	Player/People	1 day
Summary and observations taken from the previous requirements.	Observations	Players/Head Coach/Trainer/Administrator /People	3 days

[D] List Requirements

1. Security

- In the training Management System, by the conventional method, there are many concerns upon security during the transaction of fees, privacy of players and access authorization. Security is not that good in a conventional method, which can be a threat to the system. Due to the lack of security, anyone can access other person's profiles. Thus, we need to create such a system with the security of access data.

Frequency of the requirement = 3

2. Concurrent Access

- Multiple players and coaches can access multiple data and can affect multiple transactions. Schedule updating, statistics updating can be concurrent. Concurrent Access is needed to update the related data changed by the Administrator.

Frequency of the requirement = 2

3. Scheduling

- In a conventional management system, players need to depend upon a file system where schedule updating was much frequent but concurrent updating was not happening in that system.
- We need to provide a schedule of all the training sessions for the reference of trainees. We need to avoid many conflicts occurring during the scheduled training session. Sometimes we need

rescheduling of a training session, the training time of trainee with their names and need to check available slots and courts.

- As a summary, we should take below factors into consideration while scheduling,
    - Courts should not be overcrowded.
    - Number of student in one session at a time, should be less than 6.
    - Avoid Conflicts
    - Player's minimum hours and maximum hours of training

Frequency of the requirement = 7

4. Statistics

- The main motive of the system is to make the player's performance better than the previous performance.
- This can be done by the dashboard which has all the players past match/session statistics to make their game better and competitive. We need all the players' performance graded by the Trainer for each session so that Trainers can see and judge the player's strengths and weaknesses.
- This data will also help Head Couch to analyse the progress of the training academy and individual players. The annual report or monthly based report can be generated for the analysis purpose of all the players.
- As a summary we need followings for statistical purposes:
    - Skill Level of Player in each skill from skill set.
    - Overall skill level of Player.
    - Ranks of Player in individual skill and overall rank.
    - Generate Graphs based on skill levels of Players.
    - Separate page for Matches.

Frequency of the requirement = 8

5. Information

- The system needs to store all the information about player like their name, height, address, health status, default position in court and also the rating given by the trainers. The player's coach name and some information about them are also needed to be stored. Trainers' attendance also need to be store in our database as a reference purpose.
- As a summary, following are the information requirements:
    - Player's data – age, height, contact information etc.
    - Players' and Trainers' Attendance
    - Status of fees paid by Players

Frequency of the requirement = 4

6. Operating System

- Our system can be run on multiple Operating systems like Windows, MAC, android and Web. Mainly Web-Based applications can provide an easy way to check restrictions on a minimum amount of training and maximum amount of training.

Frequency of the requirement = 1

7. Scalability

- As the system is helpful and efficient we need to create a scalable system because as the number of players increases, the conventional method gets harder to manage. The system should support a large number of players and also at the same time it should be robust and efficient.

Frequency of the requirement = 1

8. Authorization

- The system needs to give access to the authorized person to prevent unintended or intended changes in the database. This is a very crucial task because it can create security risks and risks on players' details. Some authorized people like Head Coach only have access to change in players' or trainers' schedules. Players don't have any access to change these schedules by themselves.
- As a summary,
    - Separate login page for each type of user.

Frequency of the requirement = 6

9. Status

- The trainers' status about their salary needs to be updated.
- If player's fees are not paid then the state should be inactive otherwise it should be the active state of a trainee. Health status should also updated to plan the new schedule of the injured player.

Frequency of the requirement = 2

Other Requirements:

10. Data integrity

- Data integrity is needed for the overall accuracy, competencies and reliability of the data. The information needs to be error-free. The system needs to preserve error checking, validation procedures, rules and principles. It can be like a trainee cannot enter a phone number in the wrong format. The system needs to maintain a backup regularly and also need to limit access to some specific author to reduce the data integrity risk.

[E] User Classes and Characteristics

User names and basic descriptions of user roles are as follows.

1. Player

- Players will have a unique login screen on the web.
- They can log in by entering their id and password.

- Players will be able to see their training schedule – the type of training session and the name of the trainer.
- A player will receive points based on each session, players will be able to see their strengths and weaknesses – progress graph in each skill.
- Players will also have read access to available slots in case they want to request to reschedule their training session.
- Players don't have any edit access.

2. Trainer

- Trainers will have a unique login screen on the web.
- They can login by entering their id and password.
- They will give grades of players after each session, based on a scale of 1 to 10.
- Trainers will also have read access to available slots in case they want to reschedule their training session.
- Trainers are given limited read access and limited edit access, details are specified in the next parts of SRS.

3. Head Coach

- As being a Head Coach, he can access every player's and trainer's information.
- He can approve or disapprove a player's request to enter the academy.
- He is responsible for the academy's progress, thus he can monitor and track the progress of players by examining the charts and graphs generated by Basketball Training Management Application. Charts and graphs will be regarding players' attendance, grades in each skill like simple fitness, dribbling, shooting etc.
- He has the authority to add/remove training sessions for players, approve or disapprove applications regarding rescheduling training sessions from players and trainers.
- He can see what will be the number of players on courts – at what time, such that the court doesn't become overcrowded.

4. Administrator

- He is on the server-side and he doesn't have a web-based login screen.
- He can directly make changes to data, using a terminal or any other application's user interface.
- He has full authority from granting permission to removing players/trainers from a database.
- The assumption here is that, he will make changes only after discussing with Head Coach, or on the order of the head coach.

[F] Operating Environment

1. Hardware, Software or Connectivity Requirements:

- A web-based application where different user types can sign in with their credentials and get access to functions and commands as defined by the user-level privilege.

The software that is used for this application are:

DATABASE	POSTGRESQL
Front-end application	Browser-based app running on HTML and some widely supported framework linked to the database.
Customer-end application	Browser-based web-app and a smartphone (iOS/Android) application (Language used: Python)

- The required hardware and software from the user-end are:
  - Hardware: i3/i5/i7/i9 Processor or Android Version 5.0 or Higher
  - Software: Android OS or Windows 7 and up
- **Connectivity Requirements**: Wi-Fi or Mobile Internet connectivity

2. External Interface Requirements

- Third-party API: Razorpay for Payment Gateway

3. Communication Interface

- A system that allows redirection and distribution of calls and emails sent to one address/number to multiple representatives.

[G] Product Functions

Main Functions:

1. Check Available Slots

- This function can be used by all users – players, trainers, head coaches. With small differences in output, as described next.
- This will show available training session slots.
- Available training session slots are shown with the following information:

  - **Timings**
    - For example 6:00 pm to 7:00 pm.
    - After the interview with the head coach, we got to know that their training sessions are always 55 minutes and 5 minutes are given as a gap between two sessions.
    - Training sessions will always start at integer hour, more specifically available slots will have start times from the following set {6:00 AM, 7:00 AM, 8:00 AM, ……., 1:00 PM, …., 9:00 PM}. As academy remains close between 10:00 PM to 6:00 AM.

- **Courts**
  - For each available timings, available courts will be shown.

- **Training Session – Type**
  - Available Training Session Type will be shown for given Timing and Court number.
  - Training Session – Type will be from a domain: {Dribbling, Shooting, Defence, Rebounding, Passing, Fitness Session, Match Power}.
  - Match power is a special type of training session, where players are playing practise matches, and grades are given to the player based on his overall performance in a match.

- **Trainers**
  - Names of the available trainers will be shown given timing and session type. This can be got by calling isTrainerAvailable (described later) function on all trainers.

- **Players**
  - Names of the players available at that time can be got by isPlayerAvailable (described later) function on all players, with a time slot as one argument.

❖ **Differences with respect to the user:**

  - On calling this function, players can't see the available players list.
  - On calling this function, trainers can't see the available trainers list.
  - Head Coach will have access to every information

2. **Add training session of the player**

- This function will be used by Head Coach to add the player to a scheduled training session.
- Certain constraints such as restrictions on training hours and isPlayerAvailable(), will be checked before finishing this request.

3. **Delete training session of player**

- This function will be used by Head Coach to remove player from the scheduled training session.
- Certain constraints such as restrictions on training hours will be checked before finishing this request.

4. **Add training session of trainer**

- This function will be used by Head Coach to create a training session for Trainer and the head coach will also define the training session type.

5. **Delete training session of trainer**

- This function will be used by Head Coach to delete scheduled training sessions.
- On deleting session, changes will be reflected in players schedule too.

6. **Calculate Trainer's fee**

- Calculate fees (salary) of trainers, based on the number of hours and session types taught by a trainer.
- A function is used by Head Coach and Trainer. (Trainer can calculate his salary only. Whereas Head Coach can calculate any trainer's salary.)

7. **Calculate Player's fee**

- Calculate fees of players, based on the number of hours and session types attended by him.
- A function is used by Head Coach and Player. (Player can calculate his fee only. Whereas Head Coach can calculate any player's fee.)

8. **Give Grades**

- This function is used by Trainers to give grades to players, after completion of a training session.
- Head Coach can also use this, but usually, he doesn't need to.

9. **Show Progress and Attendance**

- Using this function,
    - A player can see his progress and attendance.
    - A trainer can see the progress and attendance of players he has taught in the current month.
    - Head Coach can access the information of every student

10. **Show Progress and Attendance (overall)**

- This function will be used by Head Coach only, to do the analysis of progress and attendance of the entire academy as a whole.

11. **Rank of Player**

- This function give rank of player in each skill based on average grades of grades received by player. Ranks will be given in each skill and also based on overall skill level.

12. **Separate Screen for Matches**

- This screen is accessible to all users and doesn't require login. Teams with player name and total points of the teams and schedule of matches will be visible on this screen.
- Players will enjoy playing matches and will continuously work on their skills to win the matches.

Subfunctions:

1. **Auto Status Change**:

- Change player status to inactive, if fees are not paid on time.
- This will be called directly by the software, on the checking done after each month.

2. **Approve/ Disapprove – Player's/Trainer's Admission:**

- This function is used by Head Coach, to approve or disapprove the request of the player or trainer to enter the academy.

3. **isPlayerAvailable:**

- There is the restriction on hours of training in a week, it is restricted by minimum hours of training and maximum hours of training.
- The Player can't be available if this restriction is violated or some other session is scheduled at the same time.

4. **isTrainerAvailable**:

- Trainers can't teach more than 6 students simultaneously in one session.
- Thus trainers can't be available if this restriction is violated or some other session is scheduled at the same time.

5. **isCourtAvailable**:

- Courts can't accommodate more than the certain number of players simultaneously.
- Head Coach has been instructed to set the maximum number of players at the court at one time to 25. (He also wants that this number must not be changed in future, thus changing option won't be available to all users of the application. This can only be changed by changing the code of the application).
- Thus court becomes unavailable if the number of players at one time on court reaches 25.

6. **Remove Player**:

- This function is only available to Head Coach.
- On the execution of this function, the player's data from the database will be removed, as this function will be called when players get out of the academy.

7. **Remove Trainer**:

- This function is only available to Head Coach.
- On the execution of this function, the trainer's data from the database will be removed, as this function will be called when the trainer leaves the academy.

Following is the table showing which user can access which function.

Following is the table showing which user can access which function.

☑ - full access

✓ - partial access

X – no access

FUNCTION	PLAYER	TRAINER	HEAD COACH	ADMINISTRATOR
Check Available Slots	✓	✓	☑	☑
Add training session of the player	X	X	☑	☑
Delete training session of player	X	X	☑	☑
Add training session of trainer	X	X	☑	☑
Delete training session of trainer	X	X	☑	☑
Calculate Trainers fee	X	✓	☑	☑
Calculate Player's fee	✓	X	☑	☑
Give Grades	X	✓	☑	☑
Show Progress and Attendance	✓	✓	☑	☑

[I] Assumptions

- This application is only for training and not for saving match statistics or analysis based on the match. However, matches are organized as a special training session type, and grades are given as skill – match power
- An administrator is working under the orders of the head coach. He is for technical assistance, and won't take any decision without the permission of the Head Coach.
- There isn't a limit on the number of hours a trainer can teach.
- Head Coach will be monitoring sessions, but won't be taking sessions himself.
- As trainers are getting access to the contact information of players, it is assumed that they will not start contacting students for personal coaching tuitions without the permission of the head coach and they won't to create any potential security or privacy concerns.

[J] Business Constraints

- Academy must always be in touch with the nearest hospital in case if any player got injured during the game. But Navrang Basketball Academy currently won't be able to manage this.
- Under any circumstances, if it happens to cancel the match or re-schedule then only the head coach has its authority.
- Due to budget constraints academy can't purchase high end servers. In future, with increase in number of players, they have to upgrade.
- Assigning a personal trainer to the player is very good idea to upgrade player's performance, but it is not feasible for academy right now.

Section 2

2) Noun Analysis

Problem Description

Our purpose is to provide various functionalities for the **Basketball Training Management System**. This document provides constraints, which system will follow. The final product will have features/functionalities mentioned in this document.

The software "**BASKETBALL TRAINING MANAGEMENT SYSTEM**" will be an application that will be used for managing basketball training databases. The application will manage the information about the players, the matches and the events that are to be held, coaches and the board that will plan everything. It will also store data on the physical status of all the players. This system will provide better data integrity, faster data access, minimum data inconsistency and updating data of all players and all related members. The main objective of this project is to provide a more efficient and productive way of scheduling and monitoring Basketball training sessions and help players in improving their skills.

This software is also convenient for them to announce the flow of events and other matters involved in the training sessions and practice matches held because they have the records which are essential for hosting a Basketball and they can still keep a control out of the unexpected incidents that may occur during the event. This kind of sports training management system is also very attractive in commercial sectors.

The database is about "Basketball Training Management System". This system will enable us to identify, clarify and organize relationships between various entities. This system describes how the administrator manages the player, trainer, Head Coach and other different users. Several different attributes are interrelated to each other with some relationships. The efficiency of the basketball tournament scheduling, training session, provision of schedules was the main purpose of creating this database system. This system will save schedules of all the activities with less consumption of time and effort.

For the Basketball Training Management System, we need to constantly monitor the level of skills of players. This must be the focus of our application. To accurately monitor and provide better feedback to players, we should divide training sessions into various types. Players should be graded by a trainer in each session, so that players' can get continuous reports on improvement in his level in various skills. As a training management centre, the academy should also focus on player's performance in real matches. Thus, our application also provides functionality to rate a player based on his performance in practice matches as match power skill. Trainers and Head Coach are given the right to give grades to players based on players' performance in each skill area. To analyse player's performance easily, players, trainers and Head Coach are provided with graphs on players' progress, so that they can easily track players' progress in various skills. The Head Coach is also provided with a graph on the overall skill level of the academy so that he can examine the progress of the academy as a whole.

Players are also provided with ranks in various skills based on the average of grades given to them. So that players' can get comparatively better. And players will always be motivated to continuously improve their skills to get better ranks.

Match Power skill is the most important skill that we have decided to include in the skill set of a player. As in the end performance in the match – contribution – impact made by the player, are the things which will matter. Thus the trainer will monitor players' performance in the match, and then he will give grades to the player based on his performance in the match. Other skills will be Dribbling, Shooting, Defence, Rebounding, Passing and Fitness. Players will get overall grade based on grades in each skill. Thus players will also get overall rank, based on overall grades.

Functionality to limit the number of players on a court at one time is also there. So that courts don't become overcrowded. As if the number of players on a court present at one time exceeds the certain threshold then it won't be feasible to give proper space to each player.

Academy should provide personal attention to each and every player. Personal guidance to players' is very crucial for players' progress. Thus we should limit the number of players, one trainer can teach in one session at a time.

Application will also provide a separate screen for matches' related information. List of teams (team name and team id), their total points, total matches played by them will be visible to everyone on that screen. Every match is assigned match_id, and users can see the match list - match id, team names, winner team. We should include screen related practice matches as matches are fun to play and players will get continuous motivation to improve their skills and win the matches for their team.

Other details are as follows: Players will be provided with grade cards for every month. There are different salary structures and fee structures. Player is assigned to one fee structure and trainer is assigned to one salary structure. Fee structure and salary structure are assigned based on experience of player or trainer.

Below is the basic description of the role of each type of user in the database.

Operating Environment: Web

Users:

1) **Player:**
   - The Player will have to login through the login screen by entering player id and password.
   - Players will be provided with a dashboard having a training schedule with trainers assigned to them and training session type for those sessions and a progress chart.
   - It has all the information about some personal details like the player's height, mobile number and position on the field.
   - The Dashboard will also have details about Payment Gateway, Current Application Status.
   - Players won't have additional access other than his profile.

2) **Trainer:**
   - The Trainer will have to login through the login screen by entering id and password. They all have their own unique trainer ID for login.
   - The Trainer will be provided with a dashboard having a training schedule with Players assigned for the session.
   - The Trainer will have limited read access to the contact information of players and other trainers.

3) **Head Coach:**
   - After login, the head coach has edit access to everyone's schedule, can change fees - salary structure, approve/disapprove the application of players and trainers.

4) **IT Administrator:**
   - As the admin of the system, he has full edit access and authority to grant access to others.

Brief Summary of Requirements:

- Courts should not be overcrowded.
- Number of players in one session at a time should be less than 6.
- Avoid Conflicts.
- Player's minimum hours and maximum hours of training.
- Skill Level of Player in each skill from skill set.
- Overall skill level of Player.

- Ranks of Player in individual skill and overall rank.
- Generate Graphs based on skill levels of Players.
- Separate page for Matches.
- Player's data – age, height, contact information etc.
- Players' and Trainers' Attendance.
- Status of fees paid by Players.

Below are the detailed, needed Functions:

1. **Check Available Slots**

- This function can be used by all users – players, trainers, head coaches. With small differences in output, as described next.
- This will show available training session slots.
- Available training session slots are shown with the following information:

  - **Timings**
    - For example 6:00 pm to 7:00 pm.
    - After the interview with the head coach, we got to know that their training sessions are always 55 minutes and 5 minutes are given as a gap between two sessions.
    - Training sessions will always start at integer hour, more specifically available slots will have start times from the following set {6:00 AM, 7:00 AM, 8:00 AM, ……., 1:00 PM, …., 9:00 PM}. The academy remains closed between 10:00 PM to 6:00 AM.
    - The date will be provided for all the matches for this function.

  - **Courts**
    - For each available timings, available courts will be shown.

  - **Training Session – Type**
    - Available Training Session Type will be shown for given Timing and Court number.
    - Training Session – Type will be from a domain: {Dribbling, Shooting, Defence, Rebounding, Passing, Fitness Session, Match Power}.
    - Match power is a special type of training session, where players are playing practise matches, and grades are given to the player based on his overall performance in a match.

  - **Trainers**
    - Names of the available trainers will be shown given timing and session type. This can be obtained by calling the isTrainerAvailable (described later) function on all trainers.

  - **Players**
    - Names of the players available at that time can be obtained by isPlayerAvailable (described later) function on all players, with a time slot as one argument.

  - **Differences with respect to the user:**

    - On calling this function, players can't see the available players list.
    - On calling this function, trainers can't see the available trainers list.
    - Head Coach will have access to every information

2. **Add training session of the player**

- This function will be used by the Head Coach to add the player to a scheduled training session.
- Certain constraints such as restrictions on training hours and isPlayerAvailable(), will be checked before finishing this request.

3. **Delete training session of player**

- This function will be used by Head Coach to remove players from the scheduled training session.
- Certain constraints such as restrictions on training hours will be checked before finishing this request.

4. **Add training session of trainer**

- This function will be used by Head Coach to create a training session for Trainer and the head coach will also define the training session type.

5. **Delete training session of trainer**

- This function will be used by Head Coach to delete scheduled training sessions.
- On deleting session, changes will be reflected in players schedule too.

6. **Calculate Trainer's fee**

- Calculate fees (salary) of trainers, based on the number of hours, session types and salary structure.
- There can be more than one salary structures, each trainer is assigned one salary structure
- A function is used by Head Coach and Trainer. (Trainer can calculate his salary only. Whereas the Head Coach can calculate any trainer's salary.)

7. **Calculate Player's fee**

- Calculate fees of players, based on the number of hours, session types and fee structure.
- There can be more than one fee structures, each player is assigned one fee structure.
- A function is used by Head Coach and Player. (Player can calculate his fee only. Whereas the Head Coach can calculate any player's fee.)

8. **Give Grades**

- This function is used by Trainers to give grades to players, after completion of a training session.
- The Head Coach can also use this, but usually, he doesn't need to.

9. **Show Progress and Attendance**

- Using this function,
  - A player can see his progress and attendance.
  - A trainer can see the progress and attendance of players he has taught in the current month.
  - Head Coach can access the information of every student.

10. **Show Progress and Attendance (overall)**

- This function will be used by Head Coach only, to do the analysis of progress and attendance of the entire academy as a whole.

11. **Rank of Player**

- This function gives the rank of player in each skill based on average grades of grades received by player. Ranks will be given in each skill and also based on overall skill level.

12. **Separate Screen for Matches:**

- This screen is accessible to all users and doesn't require login. Teams with player name and total points of the teams and schedule of matches will be visible on this screen.
- Players will enjoy playing matches and will continuously work on their skills to win the matches.

Subfunctions:

1. **Auto Status Change**:

- Change player status to inactive, if fees are not paid on time.
- This will be called directly by the software, on the checking done after each month.

2. **Approve/ Disapprove – Player's/Trainer's Admission:**

- This function is used by Head Coach, to approve or disapprove the request of the player or trainer to enter the academy.

3. **isPlayerAvailable:**

- There is the restriction on hours of training in a week, it is restricted by minimum hours of training and maximum hours of training.
- The Player can't be available if this restriction is violated or some other session is scheduled at the same time.

4. **isTrainerAvailable**:

- Trainers can't teach more than 6 students simultaneously in one session.
- Thus trainers can't be available if this restriction is violated or some other session is scheduled at the same time.

5. **isCourtAvailable**:

- Courts can't accommodate more than the certain number of players simultaneously.
- Head Coach has been instructed to set the maximum number of players at the court at one time to 25. (He also wants that this number must not be changed in future, thus changing option won't be available to all users of the application. This can only be changed by changing the code of the application).
- Thus court becomes unavailable if the number of players at one time on court reaches 25.

6. **Remove Player**:

- This function is only available to Head Coach.
- On the execution of this function, the player's data from the database will be removed, as this function will be called when players get out of the academy.

7. **Remove Trainer**:

- This function is only available to Head Coach.
- On the execution of this function, the trainer's data from the database will be removed, as this function will be called when the trainer leaves the academy.

Above problem description provides detailed information about all the data stored and all the functions provided by the Basketball Training Management Application.

Table 1 Noun and Verbs List

NOUN	VERB
trainer	provide
various functionalities	will
training management	follow
final product	have
software	be
basketball	used
basketball databases	manage
store data	board
physical status	plan
data integrity	access
data access	announce
minimum data inconsistency	flow
main objective	practice
productive way	can
practice matches	still
unexpected incidents	keep
commercial sectors	control
training management system	out
various entities	occur
administrator	enable
head	identify
entities	clarify
attributes	organize
basketball tournament scheduling	provision
purpose	save

database system	need
real matches	monitor
maximum hours	focus
continuous report	divide
various skills	get
management center	report
power	rate
list	match
head	right
performance	give
skill area	analyze
various skills	progress
overall skill level	track
match	graph
impact	examine
contribution	average
session id	improve
start time	include
dribbling	set
shooting	end
defense	impact
passing	matter
fitness	become
dashboard	present
rank	space
grades	teach
functionality	separate
courts	play
proper space	win
attention	grade
personal	associated
practice matches	schedule
application	profile
separate screen	id
motivation	read

below	contact
basic description	edit
environment	change
web users	approve
player’s/trainer’s admission	disapprove
session type	grant
progress chart	avoid
minimum hours	generate
approve	sets
additional access	check
login screen	output
rebounding	show
auto status	interview
contact information	got
summary	know
coach	start
head coach	close
edit access	practice
schedule	slot
salary structure	respect
full edit access	see
requirements	add
number	request
avoid conflicts	delete
minimum hours	remove
maximum hours	create
skill level	define
player	calculate
overall	use
Password	do
salary	require
height	total
position	work
mobile number	enter
player ID	accommodate

separate	option
payment	is
data – age	provides
contact information	features
status	matches
subfunctions	records
slots	sports
small	describes
differences	manages
available training session slots	minutes
timings	times
available slots	remains
pm	changes
available timings	wants
courts	becomes
type	reaches
court number	leaves
average	mentioned
month	used
special type	held
attendance	related
overall performance	involved
names	interrelated
available trainers	provided
need	motivated
time slot	decided
differences	set
certain constraints	made
delete	overcrowded
total matches	won
fee	assigned
fee structure	limited
calculate	paid
trainer’s salary	needed
disapprove	described

grades	reflected
	Pays
	taught
	Gets
	attended
	called
	restricted
	violated
	instructed
	changed
	removed
	mentioned
	used
	held
	involved
	based
	given
	provided
	paid
	restricted
	violated
	been
	instructed
	removed
	training
	managing
	updating
	hosting
	creating
	dribbling
	shooting
	rebounding
	passing
	operating
	having

	entering
	following
	timing
	playing
	calling
	finishing
	deleting
	using
	changing

Table 2 Accepted Noun and Verb list

Candidate Entity Set	**Candidate Attribute Set**	**Candidate Relationship set**
Player	• Player ID • Password, Name • height, position • mobile_number • attendance, • application_status • player_status • Birth Date • age	schedules, grades, associated, pays
Trainer	• Trainer_ID • password, name • Mobile_number • Attendance	schedules, grades, gets
Head Coach	• Name • Mobile_number	
Admin	• Name • Mobile number	
Training Session	• Session_ID • Start_Time • Date • Session_Type • Match_ID (-1 or real id)	schedule
Court	• Court_number	schedule

Team	• Team_ID • Team_Name • Team_Points • Total_Matches	Plays, associated,
Match	• Session_ID	plays
Salary Structure	• Session_Type • Salary	gets
Fee Structure	• Session_Type • Fee	pays
Grade Card	• Player_ID • Dribbling • Shooting • Defence • Rebounding • Passing • Fitness • Match_Power	grades

Table 3

Rejected Noun

NOUN	REJECT
various functionalities	General
training management	Vague
final product	General
software	General
basketball	General
basketball databases	General
store data	Vague
physical status	Attribute
data integrity	General
data access	Association
minimum data inconsistency	General
main objective	General
productive way	General
practice matches	Duplicate
unexpected incidents	Vague

commercial sectors	General
training management system	Vague
various entities	General
head	General
entities	General
attributes	General
basketball tournament scheduling	Vague
purpose	General
database system	General
real matches	Duplicate
maximum hours	Association
continuous report	General
various skills	Attribute
management center	Vague
power	Irrelevant
list	General
head	General
performance	Attribute
skill area	Irrelevant
various skills	Duplicate
overall skill level	Duplicate
impact	Vague
contribution	General
session id	Irrelevant
start time	General
dribbling	Duplicate
shooting	Duplicate
defense	Duplicate
passing	Duplicate
fitness	General
rank	Attribute
grades	General
functionality	General
courts	General
proper space	General

attention	General
personal	General
practice matches	Duplicate
application	Duplicate
separate screen	Irrelevant
motivation	General
below	Vague
basic description	General
environment	General
web users	General
player’s/trainer’s admission	Attribute
session type	General
progress chart	Association
minimum hours	Vague
approve	General
additional access	General
login screen	Irrelevant
rebounding	General
auto status	General
contact information	General
summary	Vague
coach	Irrelevant
edit access	General
schedule	Attribute
full edit access	Association
requirements	General
number	Vague
avoid conflicts	General
minimum hours	Association
skill level	Duplicate
overall	General
Password	Irrelevant
salary	Irrelevant
height	Attribute
position	Attribute

mobile number	Attribute
player ID	Attribute
separate	Irrelevant
data – age	Vague
contact information	Attribute
status	General
subfunctions	Irrelevant
Slots	General
Small	Irrelevant
differences	General
available training session slots	Association
timings	General
available slots	Duplicate
Pm	Irrelevant
available timings	Duplicate
Type	General
court number	General
Average	General
Month	General
special type	General
Attendance	Attribute
overall performance	Attribute
Names	General
available trainers	General
Need	Irrelevant
time slot	General
Differences	Irrelevant
certain constraints	General
Delete	Vague
total matches	General
fee	Attribute
calculate	General
trainer’s salary	Attribute
disapprove	General

Rejected Verb

VERB	REJECT REASON
purpose	general
follow	general
be	vague
manage	Association
plan	general
flow	irrelevant
can	vague
keep	general
out	general
enable	general
clarify	general
provision	general
monitor	general
focus	general
get	vague
rate	general
give	vague
progress	attribute
graph	general
average	general
include	vague
end	vague
matter	general
present	attribute
teach	general
win	general
read	duplicate
edit	duplicate
disapprove	general
avoid	general
check	general
output	general

interview	attribute
know	irrelevant
practice	attribute
respect	irrelevant
add	association
delete	general
create	duplicate
calculate	general
do	irrelevant
enter	general
option	attribute
features	attribute
records	general
describes	duplicate
minutes	general
remains	irrelevant
changes	duplicate
becomes	vague
leaves	general
used	vague
related	association
interrelated	association
provided	duplicate
decided	duplicate
made	general
won	attribute
limited	general
paid	attribute
described	duplicate
reflected	general
attended	general
called	general
violated	general
changed	duplicate

mentioned	general
held	general
involved	duplicate
given	duplicate
set	general
become	duplicate
paid	general
restricted	duplicate
been	irrelevant
updating	general
changing	duplicate
creating	duplicate
shooting	attribute
passing	attribute
having	irrelevant
following	duplicate
playing	duplicate
finishing	duplicate
using	general
will	vague
have	irrelevant
used	irrelevant
board	general
access	attribute
announce	attribute
practice	duplicate
still	irrelevant
control	general
occur	general
identify	general
organize	general
save	general
need	irrelevant
divide	association

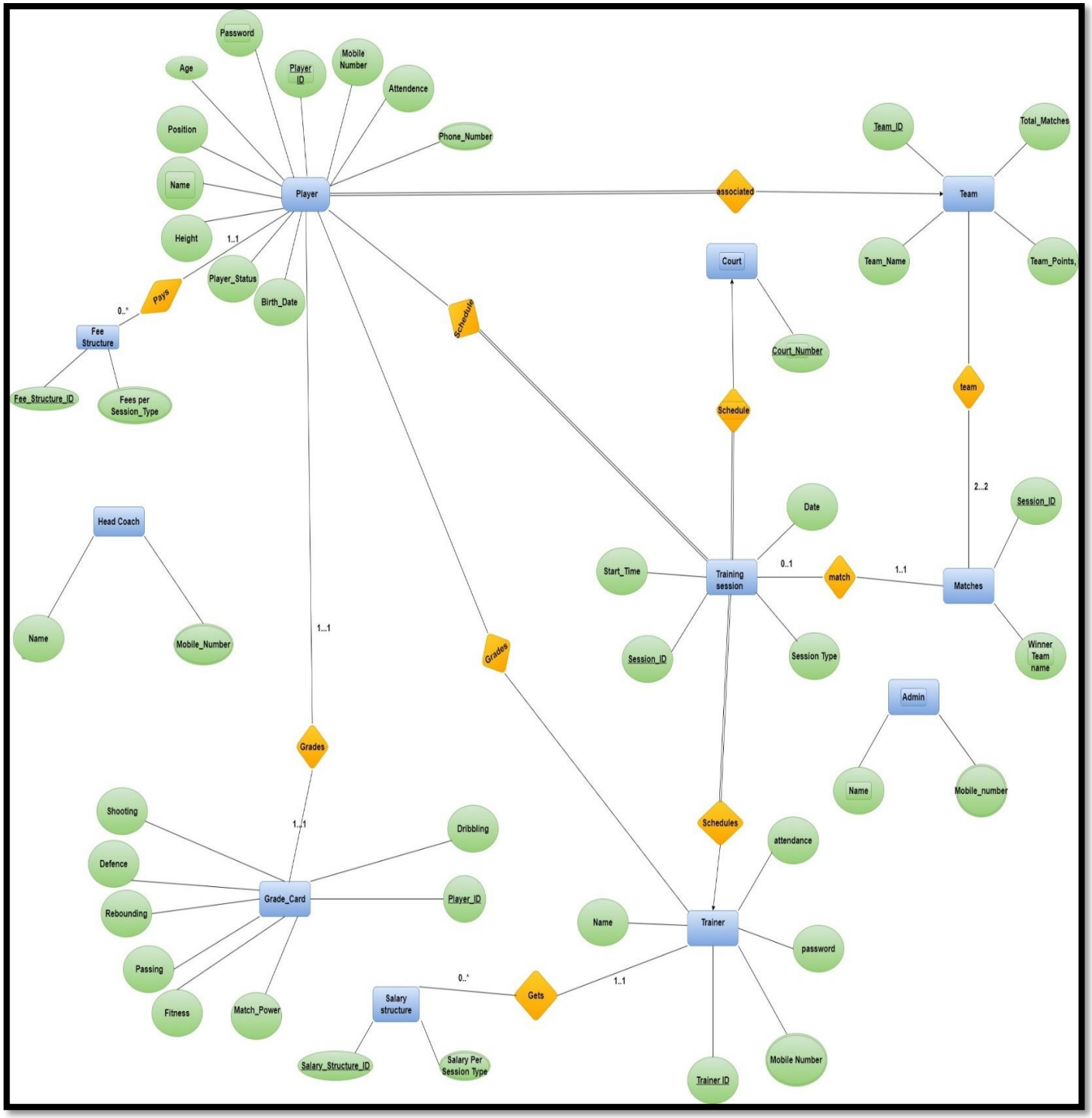
report	general
match	attribute
right	general
analyze	general
track	general
examine	attribute
impact	general
space	general
separate	general
id	attribute
contact	attribute
change	duplicate
approve	general
grant	general
generate	general
show	general
got	irrelevant
start	general
close	general
Slot	general
See	irrelevant
request	attribute
remove	general
define	general
Use	general
require	general
total	irrelevant
work	general
accommodate	general
provides	duplicate
matches	duplicate
sports	general
manages	duplicate
times	irrelevant

wants	irrelevant
reaches	general
mentioned	general
held	general
involved	general
motivated	general
set	duplicate
overcrowded	attribute
assigned	general
needed	general
taught	general
restricted	duplicate
instructed	general
removed	duplicate
used	duplicate
based	general
provided	duplicate
violated	general
instructed	general
removed	attribute
managing	general
hosting	general
dribbling	attribute
rebounding	attribute
operating	general
entering	general
timing	irrelevant
calling	general
deleting	duplicate

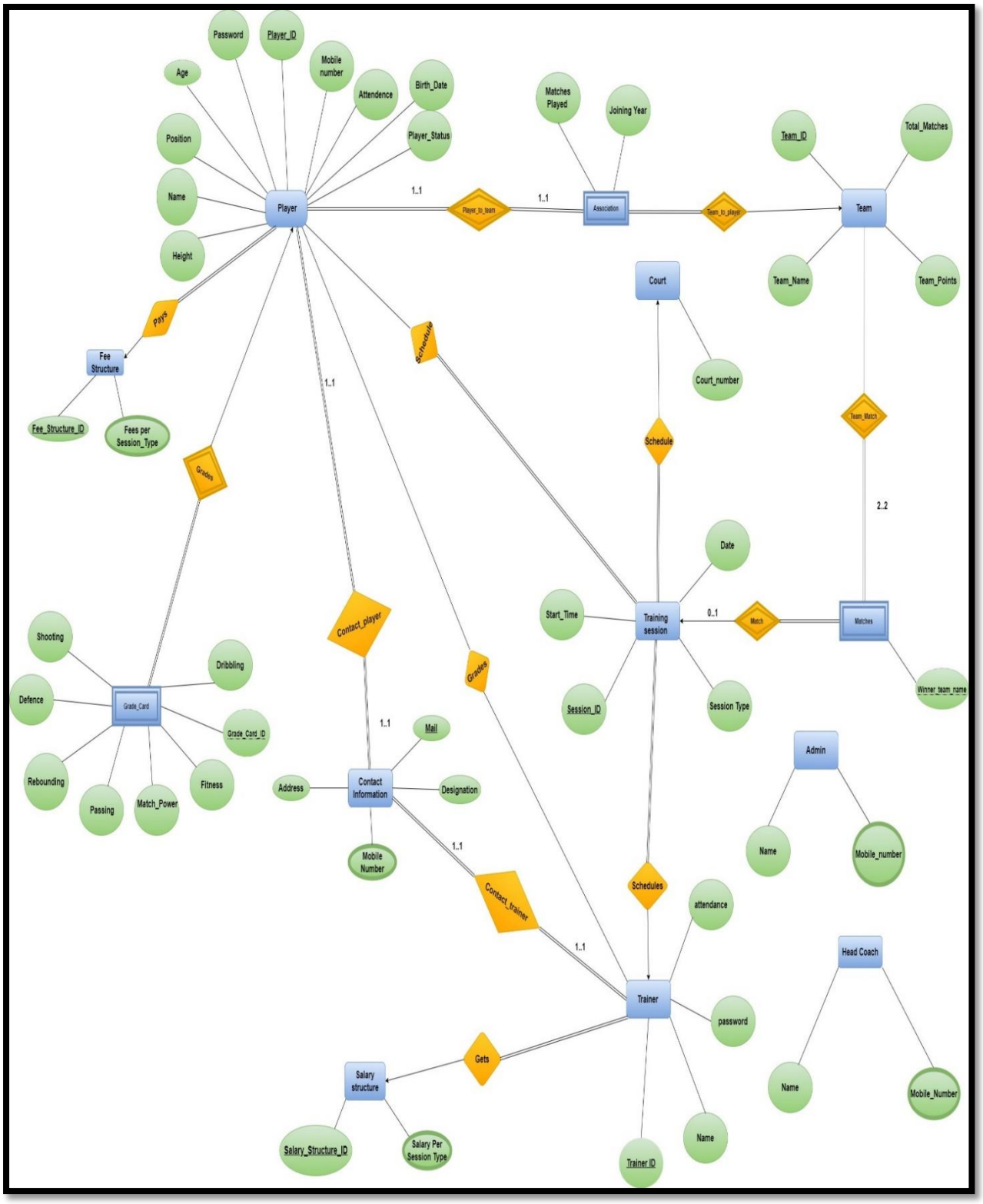
Section 3

3) ER Diagrams (all versions)

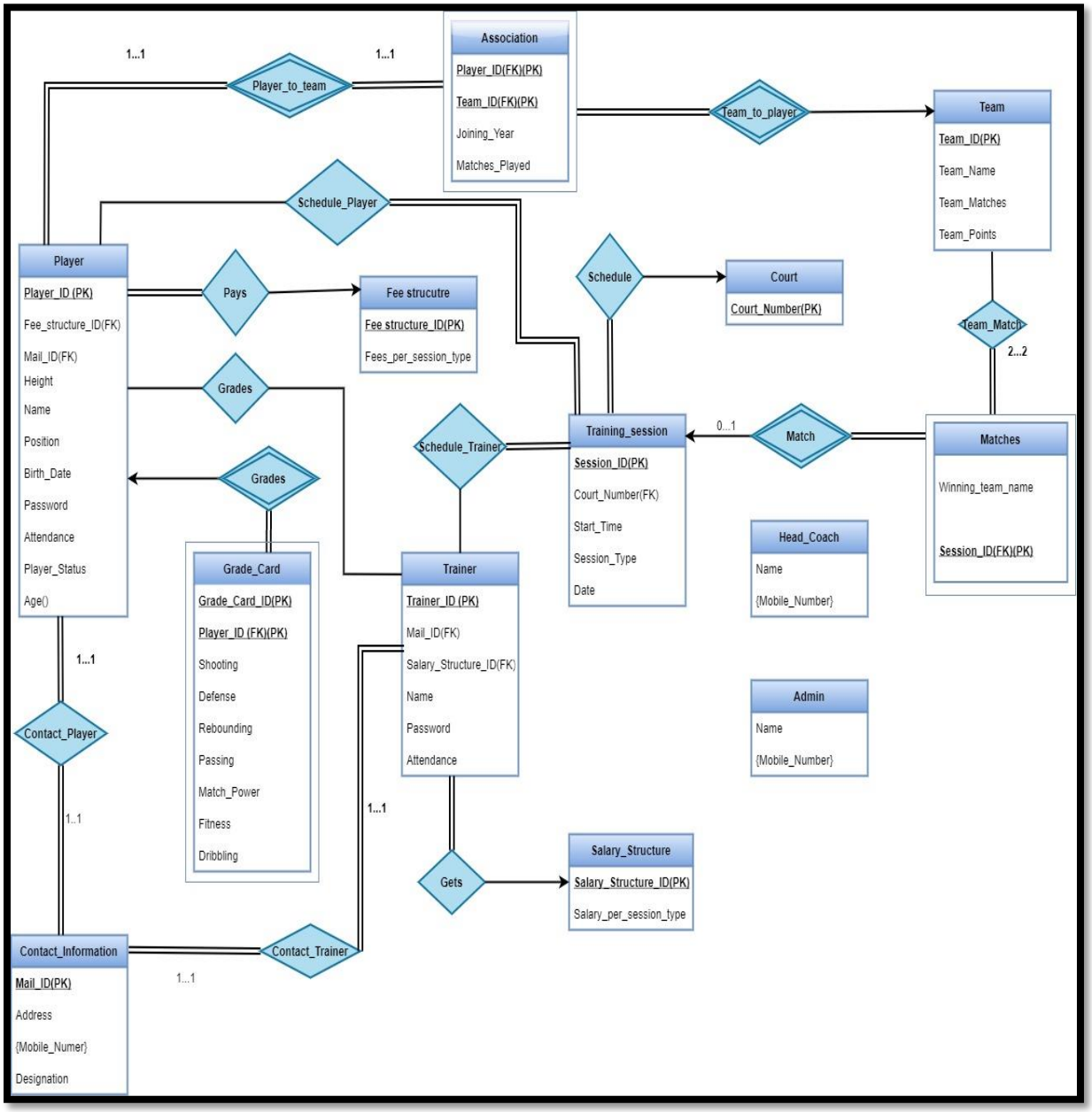
ER Diagram V1:



ER Diagram V2:



ER Diagram V3 (Final):



Section 4

4) Conversion of Final ER Diagram to Relational Model

Mapping E-R Model to Relational Model

Foreign keys are shown using (FK) in brackets.

1. **Player(Player_ID**, Fee_Structure_ID(FK), Mail_ID(FK), Height, Name, Position, Birth_Date, Password, Attendance, Player_Status, Age)
2. **Trainer(Trainer_ID,** Mail_ID(FK), Salary_structure_ID(FK),Name, Password, Attendance**)**
3. **Grade_Card(Grade_Card_ID, Player_ID(FK),** Shooting, Defence, Rebounding, Passing, Match_Power, Fitness, Dribbling**)**
4. **Contact_Information(Mail_ID,** Address, Mobile_Number, Designation**)**
5. **Matches(Session_ID(FK),**Winning_team_name**)**
6. **Salary_Structure(Salary_Structure_ID,** Salary_per_session_type**)**
7. **Training_session(Session_ID,** Court_Number(FK), Start_Time, Session_Type, Date**)**
8. **Team(Team_ID,** Team_Name, Team_Matches, Team_Points**)**
9. **Fee_Structure(Fee_structure_ID,** Fees_per_session_type**)**
10. **Admin(**Name, Mobile_Number**)**
11. **Head_Coach(**Name, Mobile_Number**)**
12. **Association(Player_ID(FK), Team_ID(FK),** Joining_year, Matches_Played**)**
13. **Court(Court_Number)**
14. **Grades(Player_ID(FK), Trainer_ID(FK))**
15. **Team_Match(Session_ID(FK), Team_ID(FK))**
16. **Schedule_Trainer(Trainer_ID(FK), Session_ID(FK))**
17. **Schedule_Player(Player_ID(FK), Session_ID(FK))**

Section 5

5) Normalization and Schema Refinement

Normalization & Schema Refinement

1. Player:

Relation:
Player(**Player_ID**, Fee_Structure_ID(FK), Mail_ID(FK), Height, Name, Position, Birth_Date, Password, Attendance, Player_Status, Age)

Dependencies:
Player_ID -> Height, Player_ID-> Name, Player_ID -> Position, Player_ID -> Birth_Date, Player_ID -> Password, Player_ID -> Attendance, Player_ID -> Player_Status, Player_ID -> Age, Player_ID -> Fee_Structure_ID, Player_ID -> Mail_ID, Mail_ID -> Player_ID, Birth_Date->Age

Primary Dependencies:
Player_ID -> Height, Player_ID-> Name, Player_ID -> Position, Player_ID -> Birth_Date, Player_ID -> Password, Player_ID -> Attendance, Player_ID -> Player_Status, Player_ID -> Age, Player_ID -> Fee_Structure_ID, Player_ID -> Mail_ID

Redundancy:

- Fee_Structure_ID will be repeated for many players, as Fee_Structure_ID is from a limited domain.
- Player_Status will be repeated, as there are costraintraint for Player_Status that can happen - {Active, Inactive, Injured}
- Position will take repeated values for players, since position in Basketball will be from - {Center,Power Forward,Small Forward,Point Guard,Shooting Guard}.

Anomalies:

Insertion: There will not be insertion anomaly, as there is no redundancy. For Player_Status during the enrollment , NULL value can be put, but in our database, we have put NOT NULL constraint in it so it will never happen.

Deletion: There will be no delete anomaly as if one wants to delete the player's information, then also no important information will be deleted as there is no important information or unnecessary deletion happens.

Update: As there is no redundancy, there won't be an update anomaly. Information can be updated without causing any undesired effect or inconsistency in the database.

1NF: Relation is already in 1NF, as there is no multivalued attribute.

2NF: Relation is already in 2NF, as 1NF is satisfied and the primary key is made up of only one attribute thus there will be no partial dependency.

3NF: Relation is already in 3NF, as 2NF is satisfied and there is no functional dependency between non-prime attributes thus there will be no transitive dependency.

BCNF: Relation is already in BCNF, as 3 NF satisfied and all the determinants are primary Key.

Note: There is transitive dependency as Birth_Date->Age, as Age is derived attribute. But as Birth_Date are not repeating in general, this will not create redundancy.

2. **Trainer:**

Relation:
Trainer(<u>Trainer_ID</u>, Mail_ID (FK), Salary_structure_ID (FK),Name, Password, Attendance**)**

Dependencies:
Trainer_ID -> Salary_structure_ID,Trainer_ID -> Name, Trainer_ID -> Password,Trainer_ID -> Attendance, Trainer_ID -> Mail_ID, Mail_ID -> Trainer_ID

Primary:
Trainer_ID -> Salary_structure_ID,Trainer_ID -> Name, Trainer_ID -> Password,Trainer_ID -> Attendance, Trainer_ID -> Mail_ID

Redundancy:
Salary structure id is also from limited domain, thus it will be repeated for most of trainers.

Anomalies:

Insertion: There will not be an insertion anomaly. For Attendance during the recruitment, NULL value can be put into it but in our database we haven't put NOT NULL constraint.

Deletion: There will be no deletion anomaly as if one wants to delete the trainer's information, then also no important information will be deleted as there is no unnecessary deletion happening.

Update: As there is no redundancy, there won't be an update anomaly. Information can be updated without causing any undesired effect or inconsistency in the database.

1NF: Relation is already in 1NF, as there is no multivalued attribute.

2NF: Relation is already in 2NF, as 1NF is satisfied and the primary key is made up of only one attribute [Trainer_ID], thus there will be no partial dependency.

3NF: Relation is already in 3NF, as 2NF is satisfied and there is no functional dependency between non-prime attributes thus there will be no transitive dependency.

BCNF: Relation is already in BCNF, as 3 NF satisfied and all the determinants are primary Key.

3. **Grade_Card:**

Relation:
Grade_Card(<u>Grade_Card_ID</u>, <u>Player_ID(FK)</u>, Shooting, Defence, Rebounding, Passing, Match_Power, Fitness, Dribbling**)**

Dependencies:
Grade_Card_ID, Player_ID -> Shooting,Grade_Card_ID, Player_ID -> Defence,Grade_Card_ID, Player_ID -> Rebounding, Passing, Grade_Card_ID, Player_ID -> Match_Power, Grade_Card_ID, Player_ID -> Fitness, Grade_Card_ID, Player_ID -> Dribbling

Primary:

Grade_Card_ID, Player_ID -> Shooting,Grade_Card_ID, Player_ID -> Defence,Grade_Card_ID, Player_ID -> Rebounding, Passing, Grade_Card_ID, Player_ID -> Match_Power, Grade_Card_ID, Player_ID -> Fitness, Grade_Card_ID, Player_ID -> Dribbling

Redundancy:
In this relation Player_ID can be repeated for many Grade_Card_ID, but as one player has many grade cards, so it will not be considered as redundancy.

Anomalies:

Insertion: The grade card of new enrolled player has all Shooting, Defence, Rebounding, Passing, Match_Power, Fitness and Dribbling as NULL value which does not create Insertion Anomaly.

Deletion: There will be no deletion anomaly as if one wants to delete the player's information about grades, then also no important information will be deleted as all other records are maintained by some other relations.

Update: As there is no redundancy, there won't be an update anomaly. Information can be updated without causing any undesired effect or inconsistency in the database.

1NF: Relation is already in 1NF, as there is no multivalued attribute.

2NF: Relation is already in 2NF, as 1NF is satisfied and relation has full dependency.

3NF: Relation is already in 3NF, as 2NF is satisfied and there is no functional dependency between non-prime attributes thus there will be no transitive dependency.

BCNF: Relation is already in BCNF, as 3 NF satisfied and all the determinants are Candidate Key.

4. **Contact_Information:**

Relation:
Contact_Information(<u>Mail_ID</u>, Address, Mobile_Number1, Mobile_Number2, Designation**)**

Dependencies:
Mail_ID -> Address, Mail_ID -> Mobile_Number1, Mail_ID -> Mobile_Number2, Mail_ID -> Designation, Mobile_Number1 -> Mail_ID, Mobile_Number2 -> Mail_ID

Primary:
Mail_ID -> Address, Mail_ID -> Mobile_Number, Mail_ID -> Designation

Redundancy:
Designation will be repeated for many players and trainers since it is from a limited domain.

Anomalies:

Insertion: There will not be insertion anomaly, as there is no redundancy. For Address , NULL value can be put into it.

Deletion: There will not be deletion anomaly.

Update: As there is no redundancy, there won't be an update anomaly. Information can be updated without causing any undesired effect or inconsistency in the database.

1NF: Relation is already in 1NF, as there is no multivalued attribute.

2NF: Relation is already in 2NF, as 1NF is satisfied and the primary key is made up of only one attribute [Mail_ID], thus there will be no partial dependency.

3NF: Relation is already in 3NF, as 2NF is satisfied and there is no functional dependency between non-prime attributes thus there will be no transitive dependency.

BCNF: Relation is already in BCNF, as 3 NF satisfied and all the determinants are primary Key or Candidate Key.

5. Matches

Relation:

Matches(<u>Session_ID(FK),</u>Winning_team_name**)**

Dependencies:
Session_ID -> Winning_team_name

Redundancy:
There will be no redundancy in this table, as there are only two attributes in the table.

Anomalies:

Insertion: There will not be insertion anomaly, as there is no redundancy. For Winning_team_name, the values will be NOT NULL.

Deletion: There will be no deletion anomaly as if you want to delete raw, then it will not affect the other relations.

Update: As there is no redundancy, there won't be an update anomaly. Information can be updated without causing any undesired effect or inconsistency in the database.

1NF: Relation is already in 1NF, as there is no multivalued attribute.

2NF: Relation is already in 2NF, as 1NF is satisfied and the primary key is made up of only one attribute [Session_ID], thus there will be no partial dependency.

3NF: Relation is already in 3NF, as 2NF is satisfied and there is no functional dependency between non-prime attributes thus there will be no transitive dependency.

BCNF: Relation is already in BCNF, as 3 NF satisfied and the determinant is a primary Key.

6. **Salary_Structure:**

Relation:

Salary_Structure(<u>Salary_Structure_ID</u>, Salary_per_session_type**)**

Dependencies:
Salary_structure_ID -> Salary_per_session_type

Redundancy:
There will be no redundancy in this table, as there are only two attributes in the table.

Anomalies:

Insertion: There will not be insertion anomaly, as there is no redundancy. For Salary_per_Session_Type, the values will be NOT NULL.

Deletion: There will be no deletion anomaly as if you want to delete raw then it will not affect the other relations.

Update: As there is no redundancy, there won't be an update anomaly. Information can be updated without causing any undesired effect or inconsistency in the database.

1NF: Relation is already in 1NF, as there is no multivalued attribute.

2NF: Relation is already in 2NF, as 1NF is satisfied and the primary key is made up of only one attribute [Salary_Structure_ID] , thus there will be no partial dependency.

3NF: Relation is already in 3NF, as 2NF is satisfied and there is no functional dependency between non-prime attributes thus there will be no transitive dependency.

BCNF: Relation is already in BCNF, as 3 NF satisfied and the determinant is a primary Key.

7. **Training_session**

Relation:

Training_session(<u>Session_ID,</u> Court_Number(FK), Start_Time, Session_Type, Date**)**

Dependencies:
Session_ID -> Start_Time, Session_ID-> Session_Type, Session_ID-> Date, Session_ID -> Court_Number, {Start_Time, Session_Type, Date, Court_Number} -> Session_ID

Primary:
Session_ID -> Winning_team_name, Session_ID -> Start_Time, Session_ID-> Session_Type, Session_ID-> Date, Session_ID -> Court_Number

Redundancy:
Court_Number and Session_Type will be repeated for many players since both are from limited domain.

Anomalies:

Insertion: There will not be insertion anomaly, as there is no redundancy. For all the attributes, values will be NOT NULL.

Deletion: There will be no deletion anomaly as if you want to delete raw then it will not affect the other relations.

Update: As there is no redundancy, there won't be an update anomaly. Information can be updated without causing any undesired effect or inconsistency in the database.

1NF: Relation is already in 1NF, as there is no multivalued attribute.

2NF: Relation is already in 2NF, as 1NF is satisfied and the primary key is made up of only one attribute [Session_ID].

3NF: Relation is already in 3NF, as 2NF is satisfied and there is no functional dependency between non-prime attributes thus there will be no transitive dependency.

BCNF: Relation is already in BCNF, as 3 NF satisfied and the determinant is a primary Key.

8. **Team**

Relation:

Team(<u>Team_ID,</u> Team_Name, Team_Matches, Team_Points**)**

Dependencies:
Team_ID -> Team_Name, Team_ID -> Team_Matches, Team_ID -> Team_Points, Team_Name->Team_ID

Primary Dependencies:
Team_ID -> Team_Name, Team_ID -> Team_Matches, Team_ID -> Team_Points

Redundancy:
Teams can be identified using Team_ID only, thus there is no database specific need for attribute Team_Name. But Team_Name is stored for the needs of the academy for creative purposes.

Anomalies:
Insertion: There will not be insertion anomaly, as there is no redundancy.

Deletion: Even if the team isn't active, there isn't any need to delete the team information. But, if one wants to then also, on the deletion only and only the team information gets deleted. Any other information like history of matches with other teams are still preserved as those are stored in different relations.

Update: As there is no redundancy, there won't be an update anomaly. Information can be updated without causing any undesired effect or inconsistency in the database.

1NF: As attributes are atomic, the table is already in 1NF.

2NF: Since the primary key contains only one attribute, there will not be any partial dependencies. Thus the relation is already in the second normal form, 2NF.

3NF: Attribute Team_Matches and Team_Points can be obtained uniquely from the knowledge of Team_Name, but as Team Names are unique, there won't be any redundancy here, thus we don't need to decompose it.

BCNF: As Team_Names are unique, Team_ID can be uniquely identified from Team_Name. Thus this creates dependency, non-prime attribute to prime attribute. But since there isn't redundancy caused, as Team_Names are unique. Thus there is no need to decompose the table.

9. **Fee_Structure**

Relation:

Fee_Structure(<u>Fee_structure_ID</u>, Fees_per_session_type**)**

Dependencies:
Fee_structure_ID -> Fees_per_session_type

Primary:
Fee_structure_ID -> Fees_per_session_type

Redundancy:
There will be no redundancy in this table, as there are only two attributes in the table.

Anomalies:
There will be no anomalies for this table as there is no redundancy in the table.

1NF: Fee_Structure_ID is stored as string and Fees_per_session_type is stored as json. Thus there is no multivalued attribute. This implies that the table is already in first normal form, 1NF.

2NF: Since, there is no partial dependency, table is already in second normal form, 2NF.

3NF: As there is only one non-prime attribute in the table, so there can't be functional dependency between two non-prime attributes. Thus table is already in third normal form, 3NF.

BCNF: Since, there is no functional dependency, where prime attribute is on right side of the dependency. Thus table is already in BCNF.

10. **Admin**

Relation:

Admin(Name, Mobile_Number1, Mobile_Number2**)**

Dependency:
Name->Mobile_Number1, Name->Mobile_Number2

Redundancy:
There won't be any redundancy in the table, as relation contains only two attributes.

Anomalies:
This is a special table, designed only for storing the information for the current admin.

Solution:
We have to modify it for storing data of previous admins.This can be done by adding two attributes namely, Joining_Date and Leaving_Date. Thus new relation is,

Admin(Name, Mobile_Number1, Mobile_Number2, Joining_Date, Leaving_Date**)**

Since, on average one data admin will work for an academy 3 to 5 years, there won't be more than 6 to 7 records in this table. Thus, there is no need for normalization for removing redundancy.

11. **Head_Coach**

Relation:

Head_Coach(Name, Mobile_Number1,Mobile_Number2**)**

Dependency:
Name->Mobile_Number1, Name->Mobile_Number2

Redundancy:
There won't be any redundancy in the table, as relation contains only two attributes.

Anomalies:
This is a special table, designed only for storing the information for the current Head Coach.

Solution:
We have to modify it for storing data of previous head coaches. This can be done by adding two attributes namely, Joining_Date and Leaving_Date. Thus new relation is,

Head_Coach(Name, Mobile_Number1,Mobile_Number2, Joining_Date, Leaving_Date**)**

Since, on average one head coach will work for an academy 3 to 5 years, there won't be more than 6 to 7 records in this table. Thus, there is no need for normalization for removing redundancy.

12. **Association**

Relation:

Association(Player_ID(FK), Team_ID(FK), Joining_year, Matches_Played**)**

Dependencies:
Player_ID, Team_ID-> Joining_year, Player_ID, Team_ID-> Matches_Played

Primary:
Player_ID, Team_ID-> Joining_year, Player_ID, Team_ID-> Matches_Played

Redundancy:
Team_ID will be repeated for many players.

Anomalies:
Insert: There aren't any anomalies associated with insert, as the table contains only and only the information of association between Player and Team.

Update: There is no way to switch teams. If team is switched by changing the Team_ID of raw, then data associated with the player and his previous team is lost.

Delete: There won't be any anomalies on delete, as the table contains only and only the information of association between Player and Team.

Solution:
An attribute, Status should be added to remove anomalies. Status should be set to 'Current' for the Player's current team, otherwise 'Past'. This way information regarding association of player and his previous team will not be lost.

Thus new relation will be,

Association(<u>Player_ID(FK)</u>, <u>Team_ID(FK)</u>, Joining_year, Matches_Played, Status**)**

Dependency:
Player_ID, Team_ID -> Joining_Year, Player_ID, Team_ID -> Matches_Played, Player_ID, Team_ID ->Status.

1NF: Since, there is no multivalued attributes, table is already in first normal form, 1NF.

2NF: Since there is no partial dependency, table is already in second normal form, 2NF.

3NF: There is no dependency between two non-prime attributes. Thus the table is already in the third normal form, 3NF.

BCNF: There is no dependency having primary attribute on right side, thus table is already in BCNF.

13. **Court**

Relation:

Court(<u>Court_Number,</u> Supervisor_Mail_ID(FK))

Since, previously information of Court Supervisors were only stored in the Contact_Information table. By adding Supervisor_Mail_ID as a foreign key, we are able to assign court specific supervisors.

Dependencies:
Court_Number -> Supervisor_Mail_ID

Primary:
Court_Number -> Supervisor_Mail_ID

Redundancy:
There won't be any redundancy in the table, as relation contains only two attributes.

Anomalies:
Since there is no redundancy, there won't be update, delete or insertion anomaly.

1NF: Since, there is no multivalued attributes, table is already in first normal form, 1NF.

2NF: Since there is no partial dependency, table is already in second normal form, 2NF.

3NF: As table contains only two attributes, there can't be dependency between two non-prime attributes. Thus the table is already in the third normal form, 3NF.

BCNF: There is no dependency having primary attribute on right side, thus table is already in BCNF.

14. **Grades(Player_ID(FK), Trainer_ID(FK))**

    This table is added as part of the converting ER diagram to Relational Model. As there was many to many relation between Player and Trainer, namely grades. So this table won't need normalization.

15. **Team_Match(Session_ID(FK), Team_ID(FK))**

    This table is added as part of the converting ER diagram to Relational Model. As there was many to many relation between Training_Session and Team, namely Team_Match. So this table won't need normalization.

16. **Schedule_Trainer(Trainer_ID(FK), Session_ID(FK))**

    This table is added as part of the converting ER diagram to Relational Model. As there was many to many relation between Trainer and Training_Session, namely Schedule_Trainer. So this table won't need normalization.

17. **Schedule_Player(Player_ID(FK), Session_ID(FK))**

    This table is added as part of the converting ER diagram to Relational Model. As there was many to many relation between Player and Training_Session, namely Schedule_Player. So this table won't need normalization.

Final Relations

1. **Player**(**Player_ID**, Fee_Structure_ID(FK), Mail_ID(FK), Height, Name, Position, Birth_Date, Password, Attendance, Player_Status, Age)
2. **Trainer(Trainer_ID,** Mail_ID (FK), Salary_structure_ID (FK),Name, Password, Attendance**)**
3. **Grade_Card(Grade_Card_ID, Player_ID(FK),** Shooting, Defence, Rebounding, Passing, Match_Power, Fitness, Dribbling**)**
4. **Contact_Information(Mail_ID,** Address, Mobile_Number1, Mobile_Number2, Designation**)**
5. **Matches(Session_ID(FK),**Winning_team_name**)**
6. **Salary_Structure(Salary_Structure_ID,** Salary_per_session_type**)**
7. **Training_session(Session_ID,** Court_Number(FK), Start_Time, Session_Type, Date**)**
8. **Team(Team_ID,** Team_Name, Team_Matches, Team_Points**)**
9. **Fee_Structure(Fee_structure_ID,** Fees_per_session_type**)**
10. **Admin(**Name, Mobile_Number1, Mobile_Number2, Joining_Date, Leaving_Date**)**
11. **Head_Coach(**Name, Mobile_Number1, Mobile_Number2, Joining_Date, Leaving_Date**)**

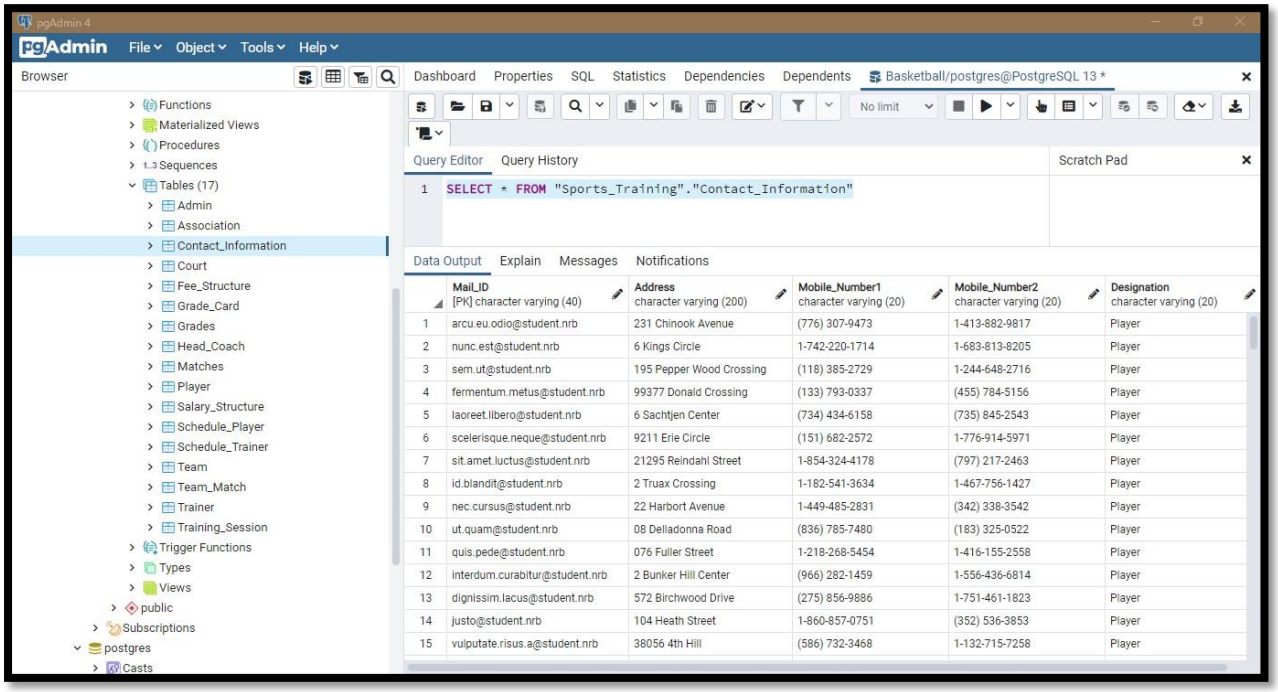
12. **Association(<u>Player_ID(FK)</u>, <u>Team_ID(FK)</u>,** Joining_year, Matches_Played, Status**)**
13. **Court(<u>Court_Number,</u>** Supervisor_Mail_ID(FK))
14. **Grades(<u>Player_ID(FK), Trainer_ID(FK)</u>)**
15. **Team_Match(<u>Session_ID(FK), Team_ID(FK)</u>)**
16. **Schedule_Trainer(<u>Trainer_ID(FK), Session_ID(FK)</u>)**
17. **Schedule_Player(<u>Player_ID(FK), Session_ID(FK)</u>)**

Section 6

6) SQL: Final DDL Scripts, Insert Statements and 40 SQL Queries with Snapshots of Output of each Query

1. **Contact Information:**

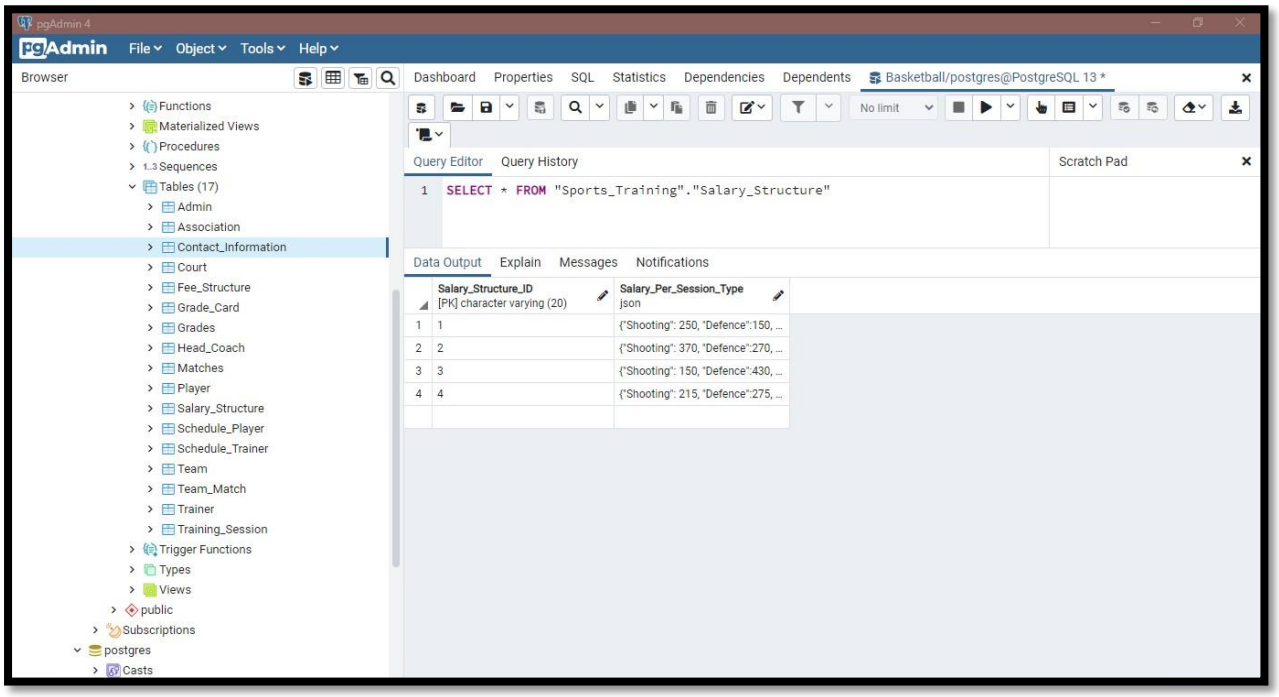
```
1.  CREATE TABLE IF NOT EXISTS "Sports_Training"."Contact_Information"
2.  (
3.      "Mail_ID" character varying(40) ,
4.      "Address" character varying(200),
5.      "Mobile_Number1" character varying(20),
6.      "Mobile_Number2" character varying(20),
7.      "Designation" character varying(20) NOT NULL,
8.      PRIMARY KEY ("Mail_ID"),
9.      CHECK("Designation" = 'Player' or "Designation" = 'Trainer' or "Designation" =
    'Court_Supervisor')
10. );
11.
12. ALTER TABLE "Sports_Training"."Contact_Information"
13.     OWNER to postgres;
14.
15. COPY "Sports_Training"."Contact_Information" FROM 'D:/Project/Contact_Information1 -
    Sheet1.csv'
16. DELIMITER ',' CSV HEADER;
```



Number of Records: 131

2. Salary_Structure:

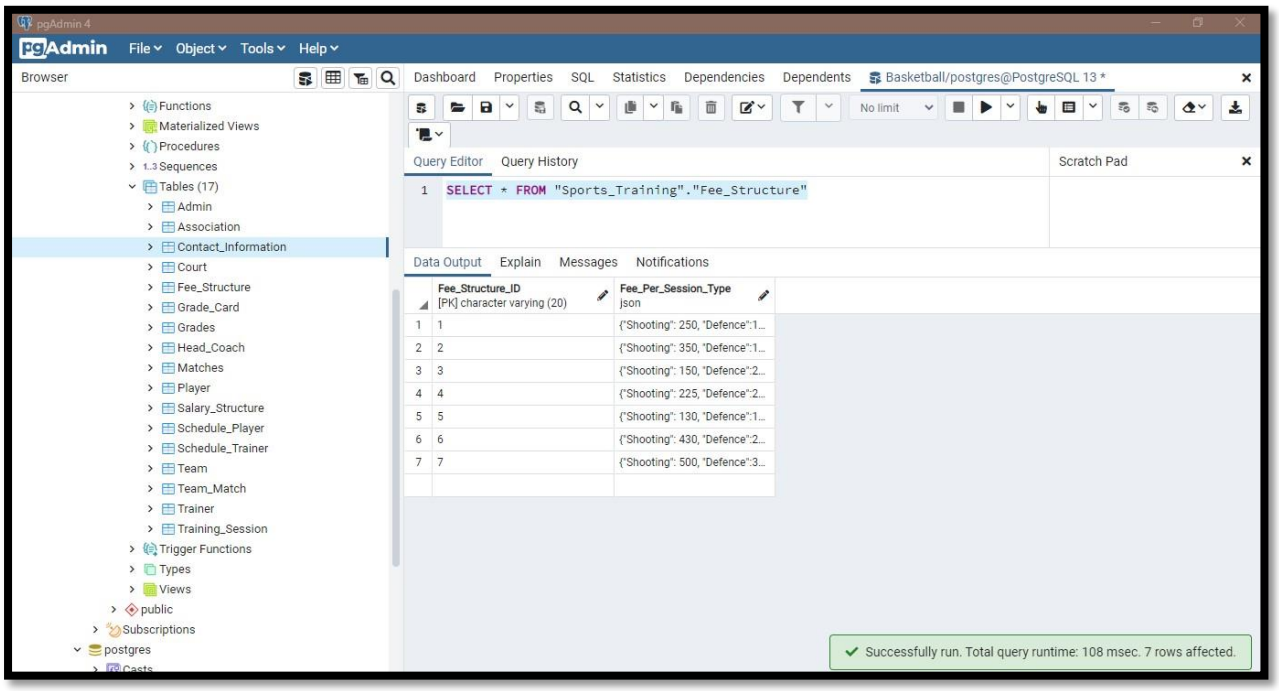
```
1.  CREATE TABLE IF NOT EXISTS "Sports_Training"."Salary_Structure"
2.  (
3.      "Salary_Structure_ID" character varying(20),
4.      "Salary_Per_Session_Type" json NOT NULL,
5.      PRIMARY KEY ("Salary_Structure_ID")
6.  );
7.
8.  ALTER TABLE "Sports_Training"."Salary_Structure"
9.      OWNER to postgres;
10.
11. COPY "Sports_Training"."Salary_Structure" FROM 'D:/Project/Salary_Structure -
    Salary_Structure.csv'
12. DELIMITER ',' CSV HEADER;
```



Number of Records: 4

3. Fee_Structure:

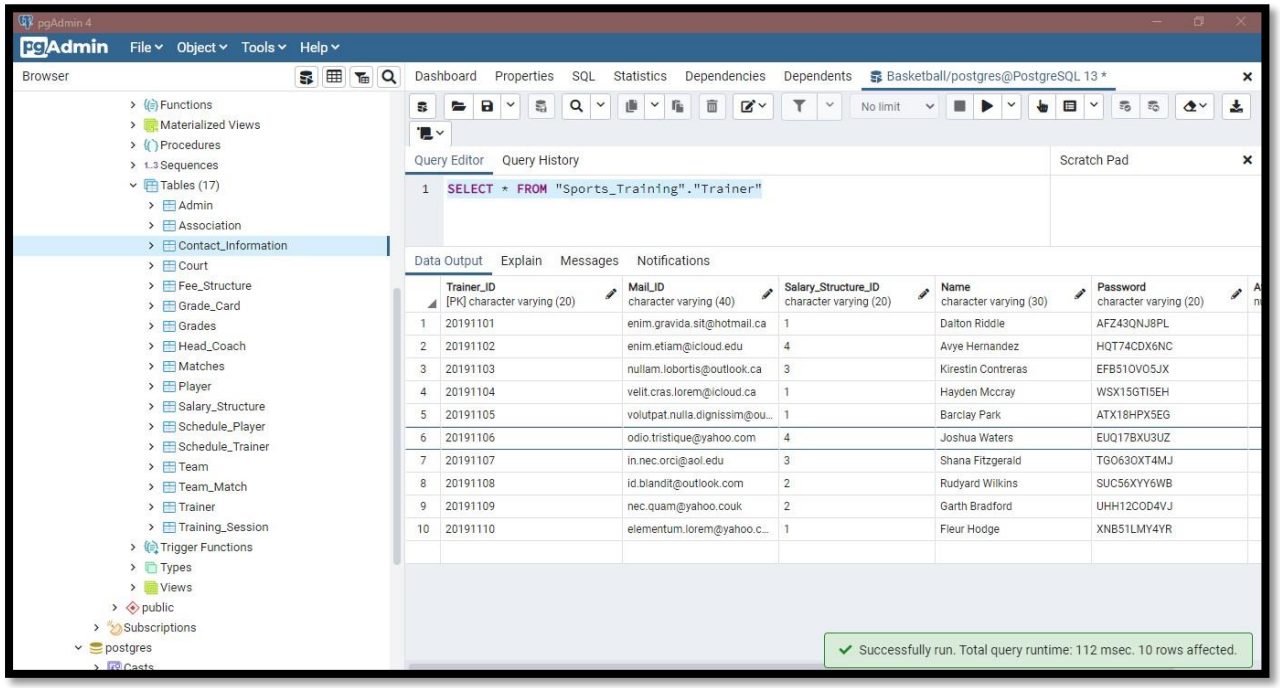
```
1.  CREATE TABLE IF NOT EXISTS "Sports_Training"."Fee_Structure"
2.  (
3.      "Fee_Structure_ID" character varying(20),
4.      "Fee_Per_Session_Type" json NOT NULL,
5.      PRIMARY KEY ("Fee_Structure_ID")
6.  );
7.
8.  ALTER TABLE "Sports_Training"."Fee_Structure"
9.      OWNER to postgres;
10.
11. COPY "Sports_Training"."Fee_Structure" FROM 'D:/Project/Fee structure.xlsx - Sheet1.csv'
12. DELIMITER ',' CSV HEADER;
```



Number of Records: 7

4. Trainer:

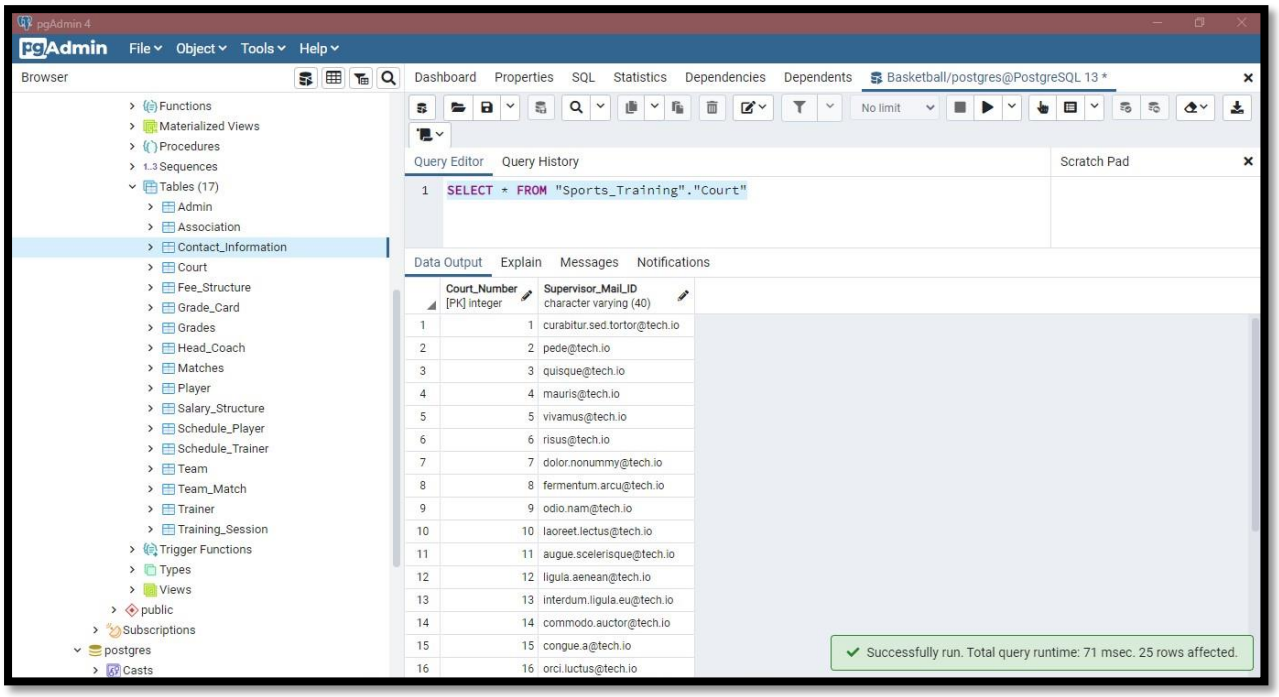
```
1.  CREATE TABLE IF NOT EXISTS "Sports_Training"."Trainer"
2.  (
3.      "Trainer_ID" character varying(20),
4.      "Mail_ID" character varying(40) NOT NULL UNIQUE,
5.      "Salary_Structure_ID" character varying(20) NOT NULL,
6.      "Name" character varying(30) NOT NULL,
7.      "Password" character varying(20) NOT NULL,
8.      "Attendance" numeric(5, 2) NOT NULL,
9.      PRIMARY KEY ("Trainer_ID"),
10.     CONSTRAINT "FK1" FOREIGN KEY ("Mail_ID")
11.         REFERENCES "Sports_Training"."Contact_Information" ("Mail_ID") MATCH SIMPLE
12.         ON UPDATE CASCADE
13.         ON DELETE RESTRICT
14.         NOT VALID,
15.     CONSTRAINT "FK2" FOREIGN KEY ("Salary_Structure_ID")
16.         REFERENCES "Sports_Training"."Salary_Structure" ("Salary_Structure_ID") MATCH SIMPLE
17.         ON UPDATE CASCADE
18.         ON DELETE RESTRICT
19.         NOT VALID
20. );
21.
22. ALTER TABLE "Sports_Training"."Trainer"
23.     OWNER to postgres;
24.
25. COPY "Sports_Training"."Trainer" FROM 'D:/Project/Trainer - Trainer.csv'
26. DELIMITER ',' CSV HEADER;
```



Number of Records: 10

5. Court:

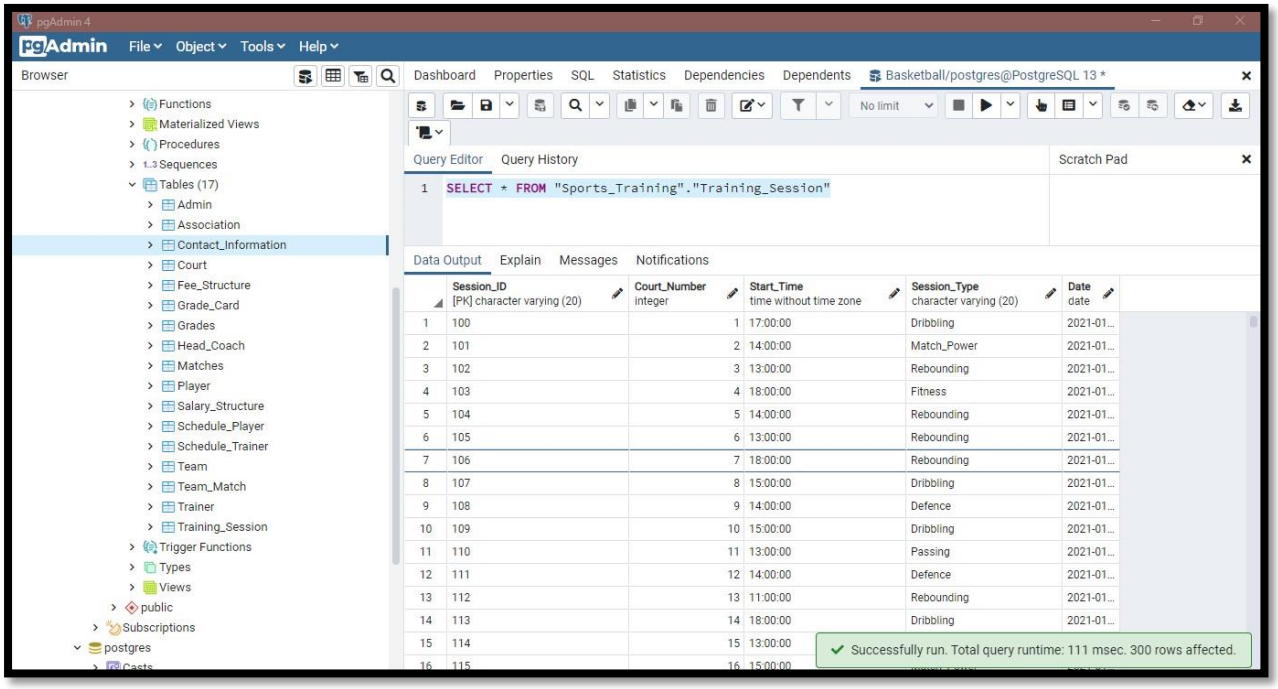
```
1.  CREATE TABLE IF NOT EXISTS "Sports_Training"."Court"
2.  (
3.      "Court_Number" integer,
4.      "Supervisor_Mail_ID" character varying(40) NOT NULL,
5.      PRIMARY KEY ("Court_Number"),
6.      CONSTRAINT "FK1" FOREIGN KEY ("Supervisor_Mail_ID")
7.          REFERENCES "Sports_Training"."Contact_Information" ("Mail_ID") MATCH SIMPLE
8.          ON UPDATE CASCADE
9.          ON DELETE RESTRICT
10.         NOT VALID
11. );
12.
13. ALTER TABLE "Sports_Training"."Court"
14.     OWNER to postgres;
15.
16. COPY "Sports_Training"."Court" FROM 'D:/Project/Court.xlsx - Sheet1.csv'
17. DELIMITER ',' CSV HEADER;
```



Number of Records: 25

6. Training_Session:

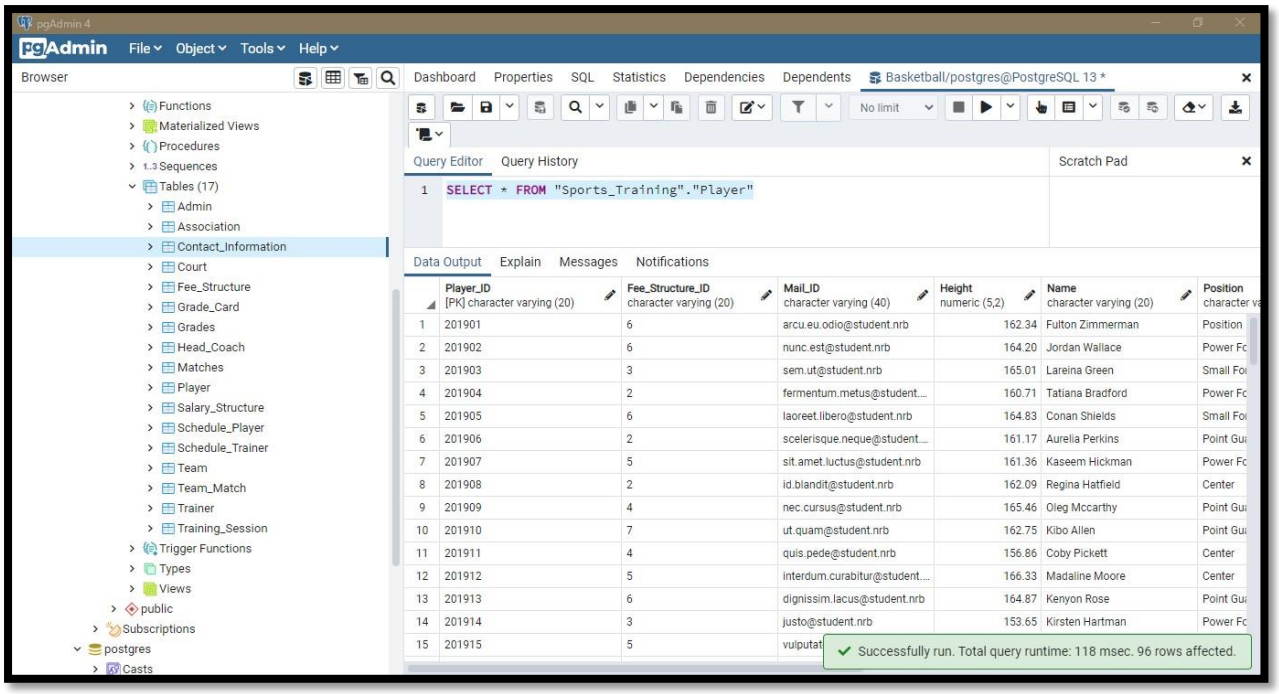
```
1.  CREATE TABLE IF NOT EXISTS "Sports_Training"."Training_Session"
2.  (
3.      "Session_ID" character varying(20),
4.      "Court_Number" integer NOT NULL,
5.      "Start_Time" time without time zone NOT NULL,
6.      "Session_Type" character varying(20) NOT NULL,
7.      "Date" date NOT NULL,
8.      PRIMARY KEY ("Session_ID"),
9.      CONSTRAINT "FK1" FOREIGN KEY ("Court_Number")
10.         REFERENCES "Sports_Training"."Court" ("Court_Number") MATCH SIMPLE
11.         ON UPDATE RESTRICT
12.         ON DELETE RESTRICT
13.         NOT VALID
14. );
15.
16. ALTER TABLE "Sports_Training"."Training_Session"
17.     OWNER to postgres;
18.
19. COPY "Sports_Training"."Training_Session" FROM 'D:/Project/Training_session.xlsx - Sheet1.csv'
20. DELIMITER ',' CSV HEADER;
```



Number of Records: 300

7. Player:

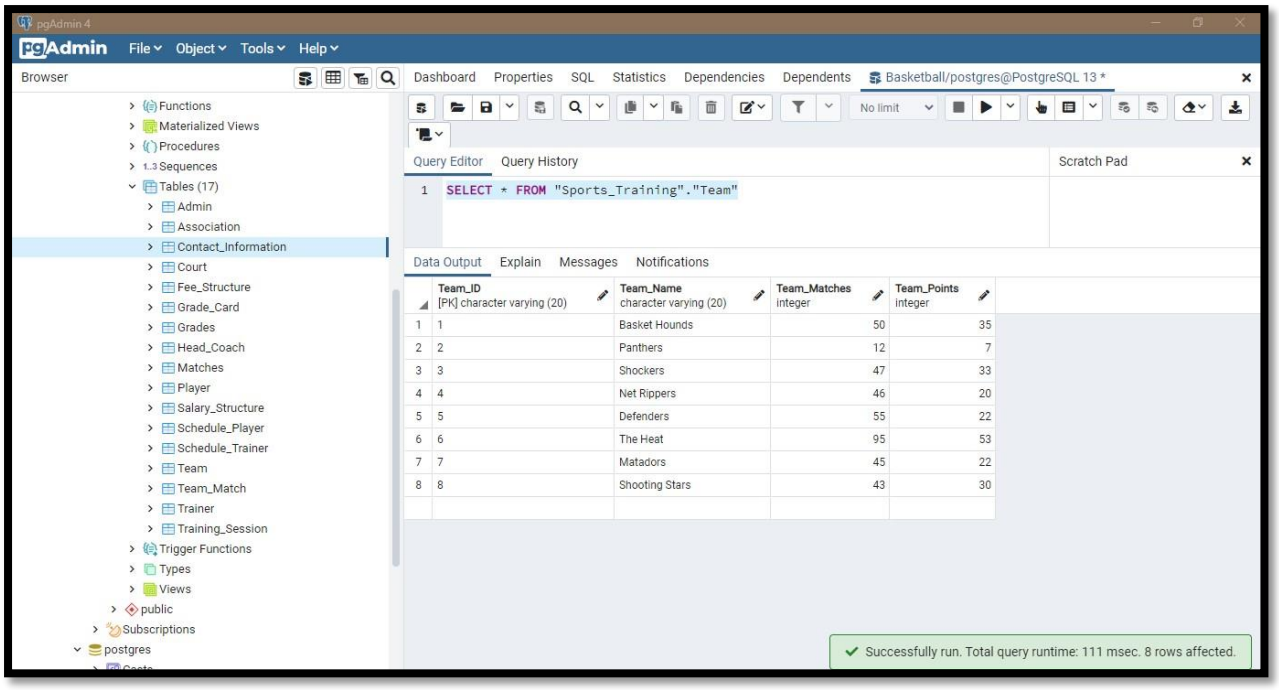
```
CREATE TABLE IF NOT EXISTS "Sports_Training"."Player"
(
    "Player_ID" character varying(20),
    "Fee_Structure_ID" character varying(20) NOT NULL,
    "Mail_ID" character varying(40) NOT NULL UNIQUE,
    "Height" numeric(5, 2) NOT NULL,
    "Name" character varying(20) NOT NULL,
    "Position" character varying(20) NOT NULL,
    "Birth_Date" date NOT NULL,
    "Password" character varying(20) NOT NULL,
    "Attendance" numeric(5, 2) NOT NULL,
    "Player_Status" character varying(20) NOT NULL,
    "Age" integer,
    PRIMARY KEY ("Player_ID"),
    CONSTRAINT "Fk1" FOREIGN KEY ("Fee_Structure_ID")
        REFERENCES "Sports_Training"."Fee_Structure" ("Fee_Structure_ID") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE RESTRICT
        NOT VALID,
    CONSTRAINT "FK2" FOREIGN KEY ("Mail_ID")
        REFERENCES "Sports_Training"."Contact_Information" ("Mail_ID") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE RESTRICT
        NOT VALID,
   CHECK("Player_Status" = 'Active' or "Player_Status" = 'Inactive' or "Player_Status" = 'Injured')
);

ALTER TABLE "Sports_Training"."Player"
    OWNER to postgres;

COPY "Sports_Training"."Player" FROM 'D:/Project/Player - Player.csv'
DELIMITER ',' CSV HEADER;
```



Number of Records: 96

8. Team:

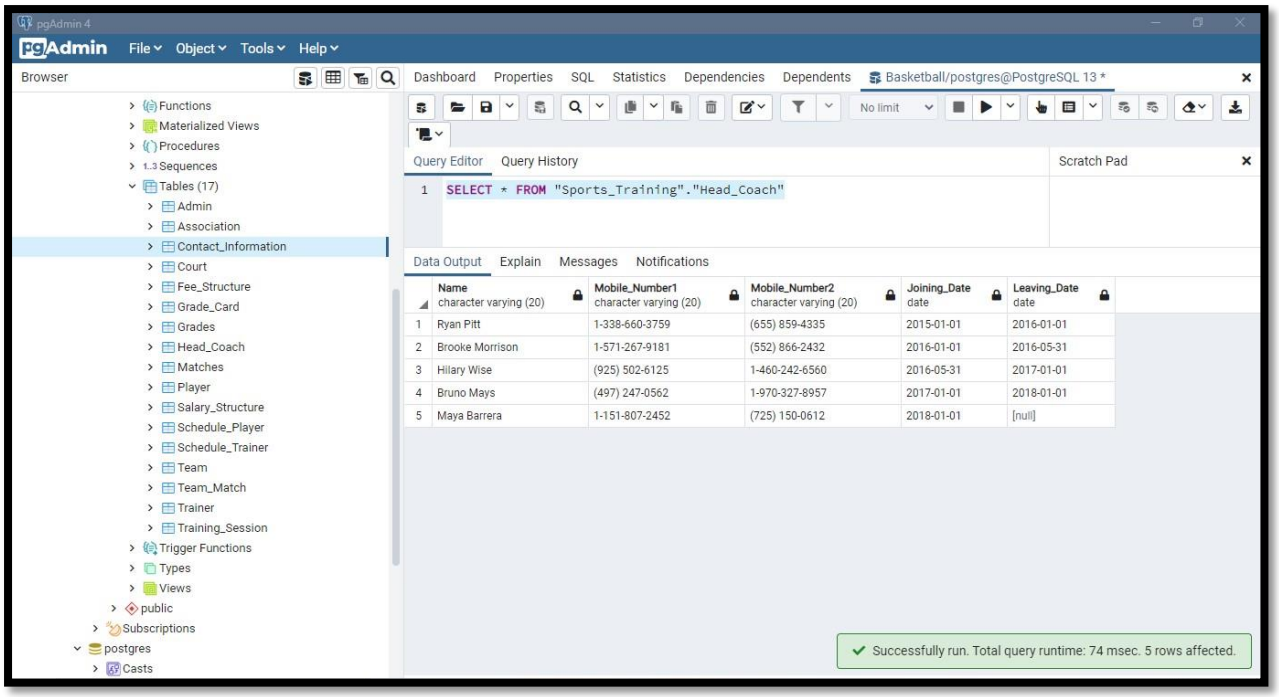
```
1.  CREATE TABLE IF NOT EXISTS "Sports_Training"."Team"
2.  (
3.      "Team_ID" character varying(20),
4.      "Team_Name" character varying(20) NOT NULL,
5.      "Team_Matches" integer NOT NULL,
6.      "Team_Points" integer NOT NULL,
7.      PRIMARY KEY ("Team_ID")
8.  );
9.
10. ALTER TABLE "Sports_Training"."Team"
11.     OWNER to postgres;
12.
13. COPY "Sports_Training"."Team" FROM 'D:/Project/Team.xlsx - Sheet1.csv'
14. DELIMITER ',' CSV HEADER;
```



Number of Records: 8

9. Head Coach:

```
1.  CREATE TABLE IF NOT EXISTS "Sports_Training"."Head_Coach"
2.  (
3.      "Name" character varying(20) NOT NULL,
4.      "Mobile_Number1" character varying(20),
5.      "Mobile_Number2" character varying(20),
6.       "Joining_Date" date NOT NULL,
7.      "Leaving_Date" date
8.  );
9.
10. ALTER TABLE "Sports_Training"."Head_Coach"
11.     OWNER to postgres;
12.
13. COPY "Sports_Training"."Head_Coach" FROM 'D:/Project/Head Coach.xlsx - Sheet1.csv'
14. DELIMITER ',' CSV HEADER;
```



Number of Records: 5

10. **Admin:**

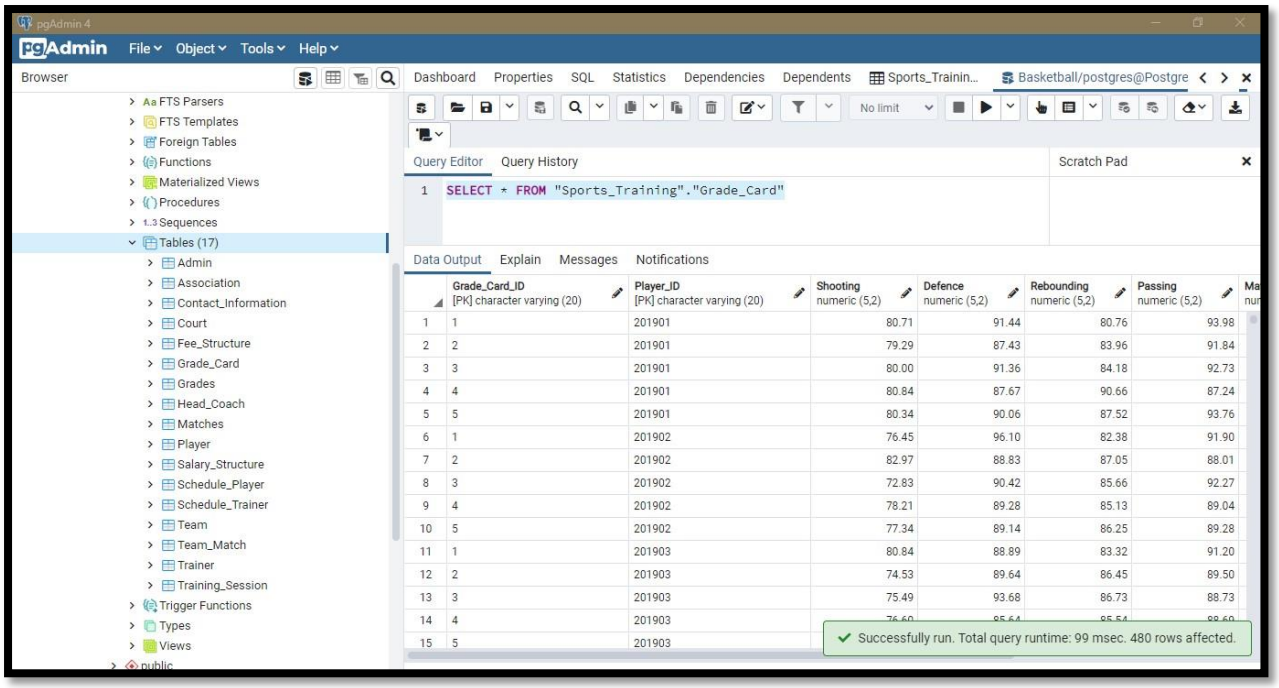
```
1.  CREATE TABLE IF NOT EXISTS "Sports_Training"."Admin"
2.  (
3.      "Name" character varying(20) NOT NULL,
4.      "Mobile_Number1" character varying(20),
5.      "Mobile_Number2" character varying(20),
6.      "Joining_Date" date NOT NULL,
7.      "Leaving_Date" date
8.  );
9.
10. ALTER TABLE "Sports_Training"."Admin"
11.     OWNER to postgres;
12.
13. COPY "Sports_Training"."Admin" FROM 'D:/Project/Admin - Admin.csv'
14. DELIMITER ',' CSV HEADER;
```

Number of Records: 7

11. **Grade_Card:**

```
1.  CREATE TABLE IF NOT EXISTS "Sports_Training"."Grade_Card"
2.  (
3.      "Grade_Card_ID" character varying(20) NOT NULL,
4.      "Player_ID" character varying(20) NOT NULL,
```

```
    "Shooting" numeric(5, 2),
    "Defence" numeric(5, 2),
    "Rebounding" numeric(5, 2),
    "Passing" numeric(5, 2),
    "Match_Power" numeric(5, 2),
    "Fitness" numeric(5, 2),
    "Dribbling" numeric(5, 2),
    PRIMARY KEY ("Grade_Card_ID", "Player_ID"),
    CONSTRAINT "FK1" FOREIGN KEY ("Player_ID")
        REFERENCES "Sports_Training"."Player" ("Player_ID") MATCH SIMPLE
        ON UPDATE RESTRICT
        ON DELETE CASCADE
        NOT VALID
);

ALTER TABLE "Sports_Training"."Grade_Card"
    OWNER to postgres;

COPY "Sports_Training"."Grade_Card" FROM 'D:/Project/Grade_Card - Sheet1.csv'
DELIMITER ',' CSV HEADER;
```

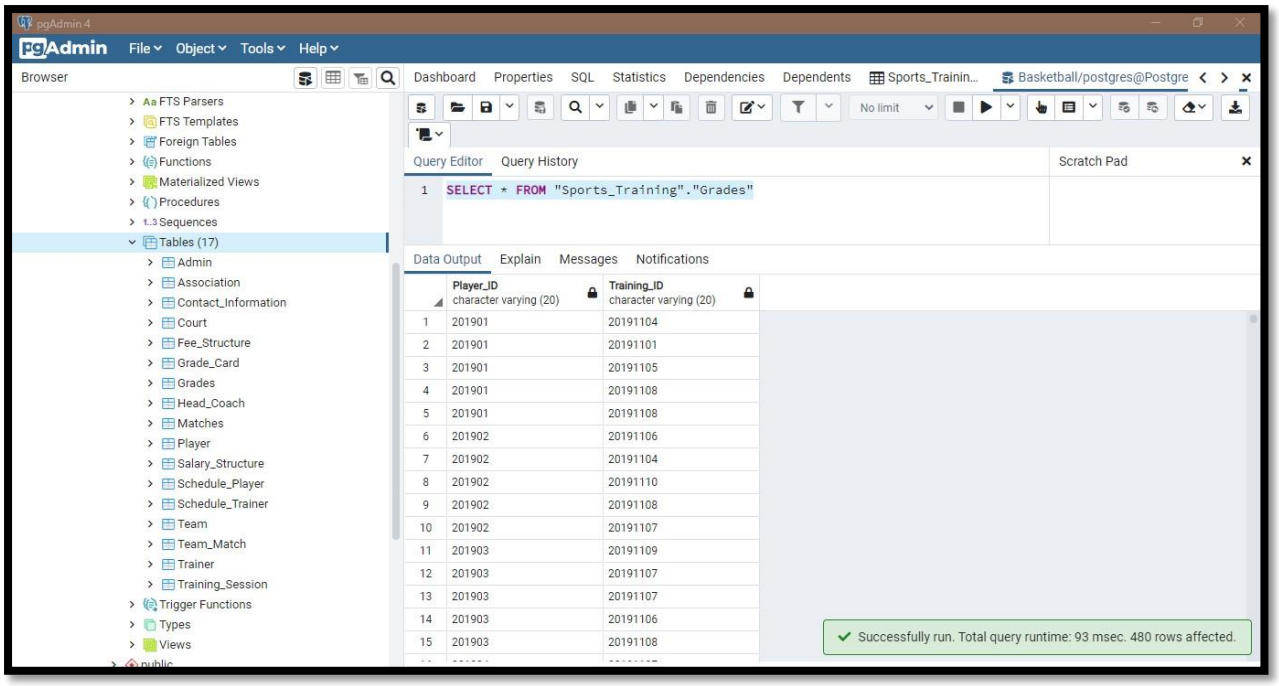


Number of Records: 480

12. Grades:

```
CREATE TABLE IF NOT EXISTS "Sports_Training"."Grades"
(
    "Player_ID" character varying(20) NOT NULL,
    "Training_ID" character varying(20) NOT NULL,
```

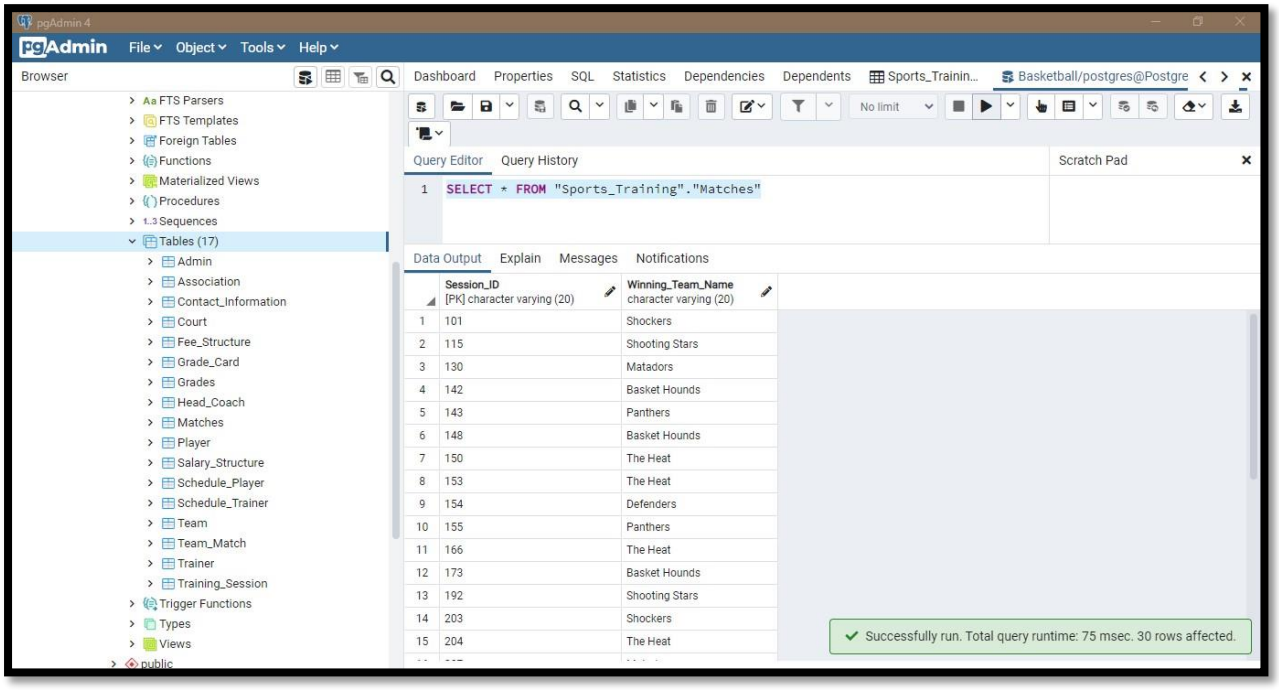
```
    CONSTRAINT "FK1" FOREIGN KEY ("Player_ID")
        REFERENCES "Sports_Training"."Player" ("Player_ID") MATCH SIMPLE
        ON UPDATE RESTRICT
        ON DELETE CASCADE
        NOT VALID,
    CONSTRAINT "FK2" FOREIGN KEY ("Training_ID")
        REFERENCES "Sports_Training"."Trainer" ("Trainer_ID") MATCH SIMPLE
        ON UPDATE RESTRICT
        ON DELETE CASCADE
        NOT VALID
);

ALTER TABLE "Sports_Training"."Grades"
    OWNER to postgres;

COPY "Sports_Training"."Grades" FROM 'D:/Project/Grades - Sheet1.csv'
DELIMITER ',' CSV HEADER;
```



Number of Records: 480

13. **Matches:**

```
1.  CREATE TABLE IF NOT EXISTS "Sports_Training"."Matches"
2.  (
3.      "Session_ID" character varying(20),
4.      "Winning_Team_Name" character varying(20) NOT NULL,
5.      PRIMARY KEY ("Session_ID"),
6.      CONSTRAINT "FK1" FOREIGN KEY ("Session_ID")
7.          REFERENCES "Sports_Training"."Training_Session" ("Session_ID") MATCH SIMPLE
8.          ON UPDATE RESTRICT
9.          ON DELETE CASCADE
10.         NOT VALID
11. );
12.
13. ALTER TABLE "Sports_Training"."Matches"
14.     OWNER to postgres;
15.
16. COPY "Sports_Training"."Matches" FROM 'D:/Project/Matches - Sheet1.csv'
17. DELIMITER ',' CSV HEADER;
```

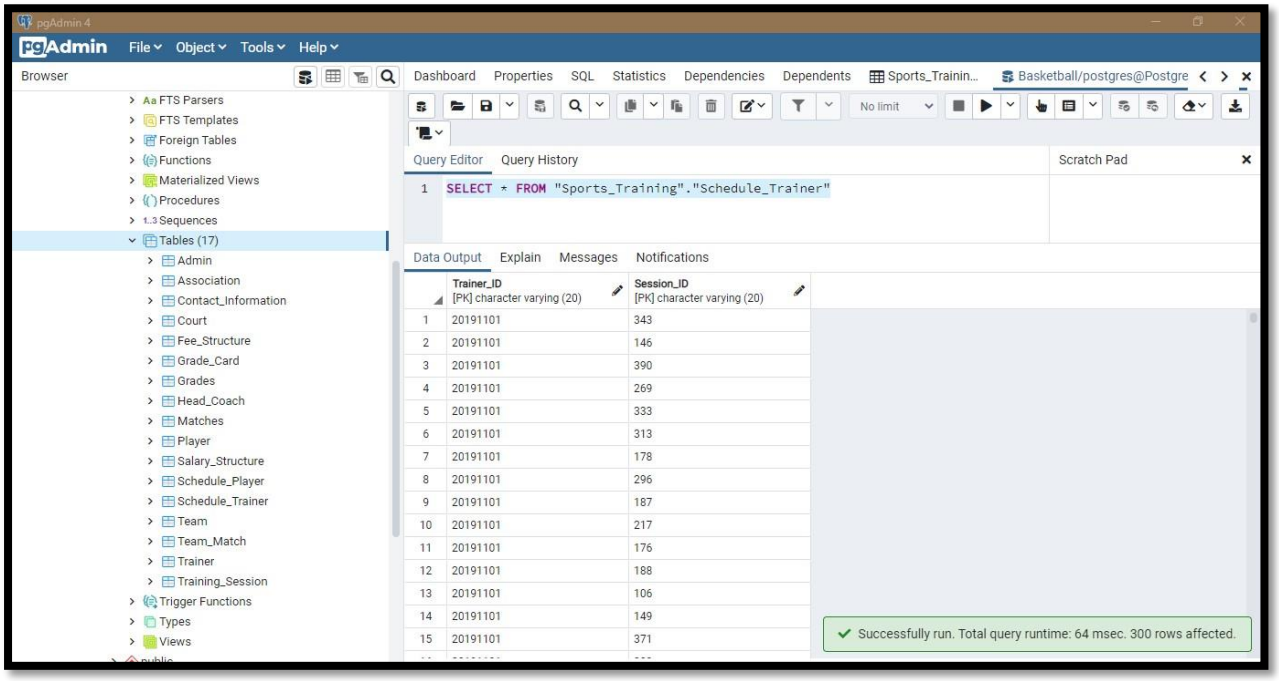


Number of Records:30

14. **Schedule_Trainer:**

```
1.  CREATE TABLE IF NOT EXISTS "Sports_Training"."Schedule_Trainer"
2.  (
3.      "Trainer_ID" character varying(20) NOT NULL,
4.      "Session_ID" character varying(20) NOT NULL,
```

```
    PRIMARY KEY ("Trainer_ID", "Session_ID"),
    CONSTRAINT "FK1" FOREIGN KEY ("Trainer_ID")
      REFERENCES "Sports_Training"."Trainer" ("Trainer_ID") MATCH SIMPLE
      ON UPDATE RESTRICT
      ON DELETE CASCADE
      NOT VALID,
       CONSTRAINT "FK2" FOREIGN KEY ("Session_ID")
         REFERENCES "Sports_Training"."Training_Session" ("Session_ID")
         ON UPDATE RESTRICT
         ON DELETE CASCADE
         NOT VALID
);

ALTER TABLE "Sports_Training"."Schedule_Trainer"
    OWNER to postgres;

COPY "Sports_Training"."Schedule_Trainer" FROM 'D:/Project/Schedule_Trainer - Sheet1.csv'
DELIMITER ',' CSV HEADER;
```

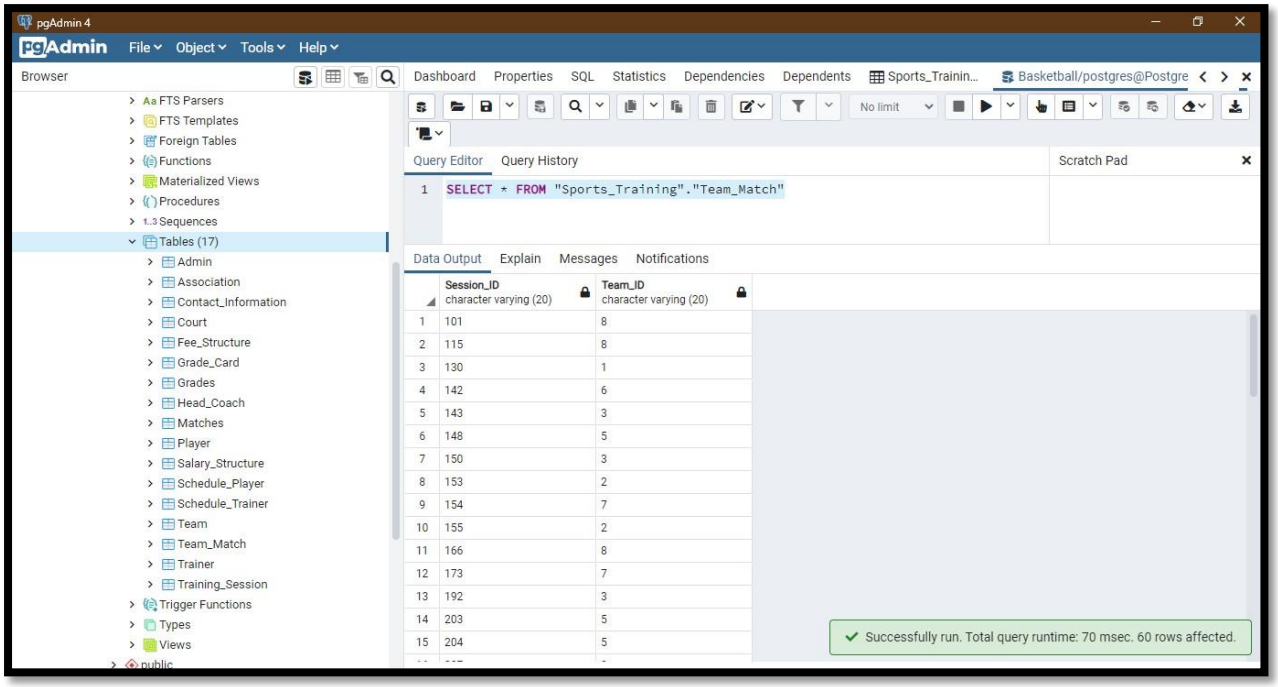


Number of Records: 300

15. Team_Match:

```
CREATE TABLE IF NOT EXISTS "Sports_Training"."Team_Match"
(
    "Session_ID" character varying(20) NOT NULL,
    "Team_ID" character varying(20) NOT NULL,
    CONSTRAINT "FK1" FOREIGN KEY ("Session_ID")
      REFERENCES "Sports_Training"."Matches" ("Session_ID") MATCH SIMPLE
      ON UPDATE RESTRICT
```

```
        ON DELETE CASCADE
        NOT VALID,
    CONSTRAINT "FK2" FOREIGN KEY ("Team_ID")
        REFERENCES "Sports_Training"."Team" ("Team_ID") MATCH SIMPLE
        ON UPDATE RESTRICT
        ON DELETE CASCADE
        NOT VALID
);

ALTER TABLE "Sports_Training"."Team_Match"
    OWNER to postgres;

COPY "Sports_Training"."Team_Match" FROM 'D:/Project/Team_Match - Sheet1.csv'
DELIMITER ',' CSV HEADER;
```

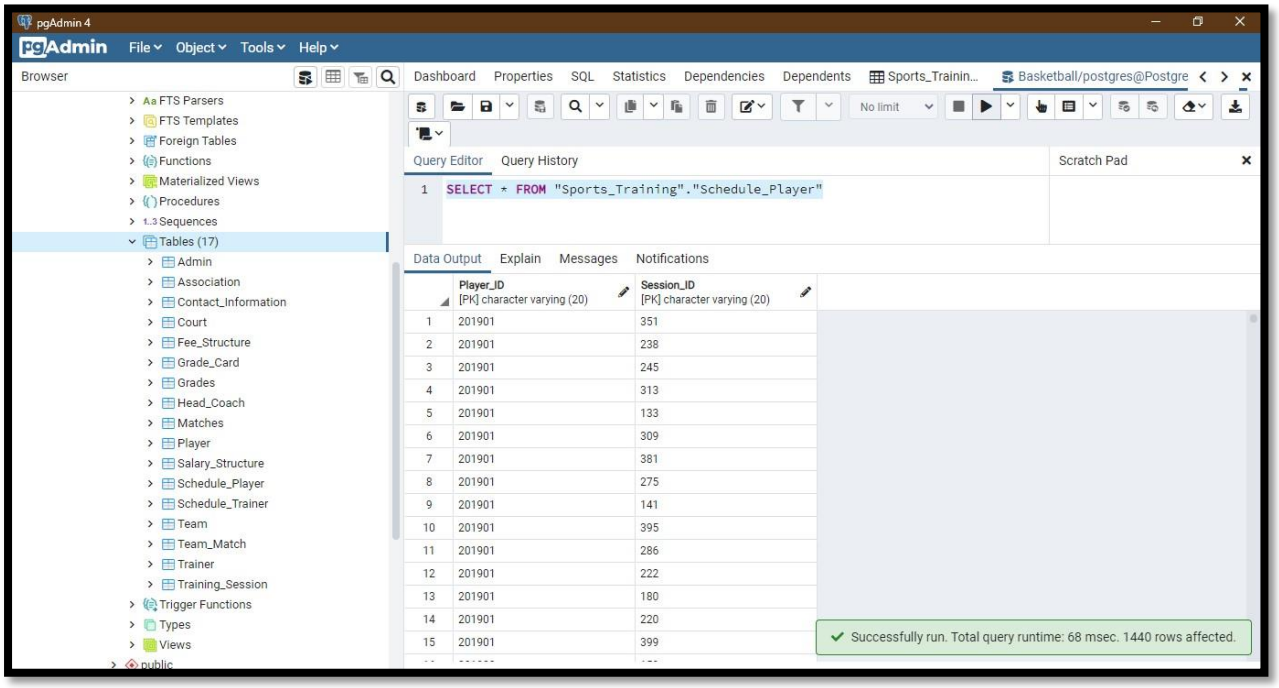


Number of Records: 60

16. Schedule_Player:

```
CREATE TABLE IF NOT EXISTS "Sports_Training"."Schedule_Player"
(
    "Player_ID" character varying(20) NOT NULL,
    "Session_ID" character varying(20) NOT NULL,
    PRIMARY KEY ("Player_ID", "Session_ID"),
    CONSTRAINT "FK1" FOREIGN KEY ("Player_ID")
        REFERENCES "Sports_Training"."Player" ("Player_ID") MATCH SIMPLE
        ON UPDATE RESTRICT
        ON DELETE CASCADE
        NOT VALID,
    CONSTRAINT "FK2" FOREIGN KEY ("Session_ID")
```

```
12.         REFERENCES "Sports_Training"."Training_Session" ("Session_ID")
13.         ON UPDATE RESTRICT
14.         ON DELETE CASCADE
15.         NOT VALID
16.  );
17.
18.  ALTER TABLE "Sports_Training"."Schedule_Player"
19.      OWNER to postgres;
20.
21.  COPY "Sports_Training"."Schedule_Player" FROM 'D:/Project/Schedule_Player - Sheet1.csv'
22.  DELIMITER ',' CSV HEADER;
```

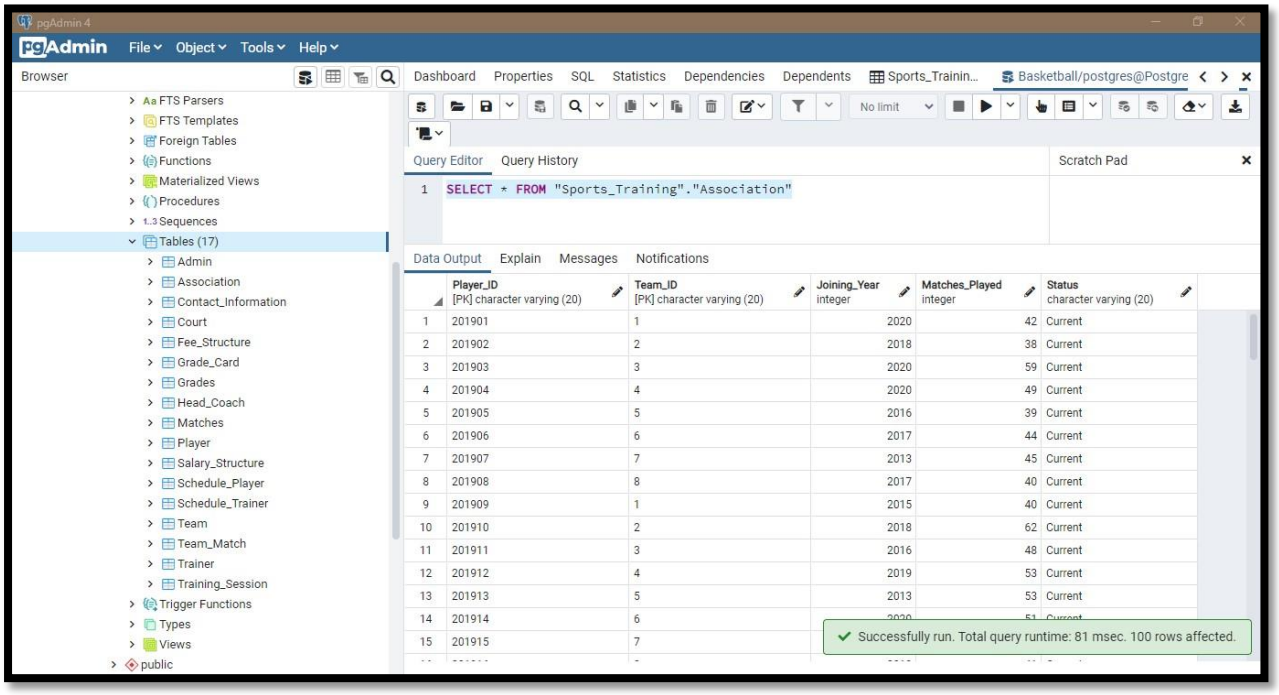


Number of Records: 1440

17. Association:

```
1.  CREATE TABLE IF NOT EXISTS "Sports_Training"."Association"
2.  (
3.      "Player_ID" character varying(20) NOT NULL,
4.      "Team_ID" character varying(20) NOT NULL,
5.      "Joining_Year" integer NOT NULL,
6.      "Matches_Played" integer,
7.      "Status" character varying(20) NOT NULL,
8.      PRIMARY KEY ("Player_ID", "Team_ID"),
9.      CONSTRAINT "FK1" FOREIGN KEY ("Player_ID")
10.         REFERENCES "Sports_Training"."Player" ("Player_ID") MATCH SIMPLE
11.         ON UPDATE RESTRICT
12.         ON DELETE CASCADE
13.         NOT VALID,
14.     CONSTRAINT "FK2" FOREIGN KEY ("Team_ID")
```

```
15.         REFERENCES "Sports_Training"."Team" ("Team_ID") MATCH SIMPLE
16.         ON UPDATE RESTRICT
17.         ON DELETE CASCADE
18.         NOT VALID
19.  );
20.
21.  ALTER TABLE "Sports_Training"."Association"
22.      OWNER to postgres;
23.
24.  COPY "Sports_Training"."Association" FROM 'D:/Project/Association - Sheet1.csv'
25.  DELIMITER ',' CSV HEADER;
```



Number of Records: 100

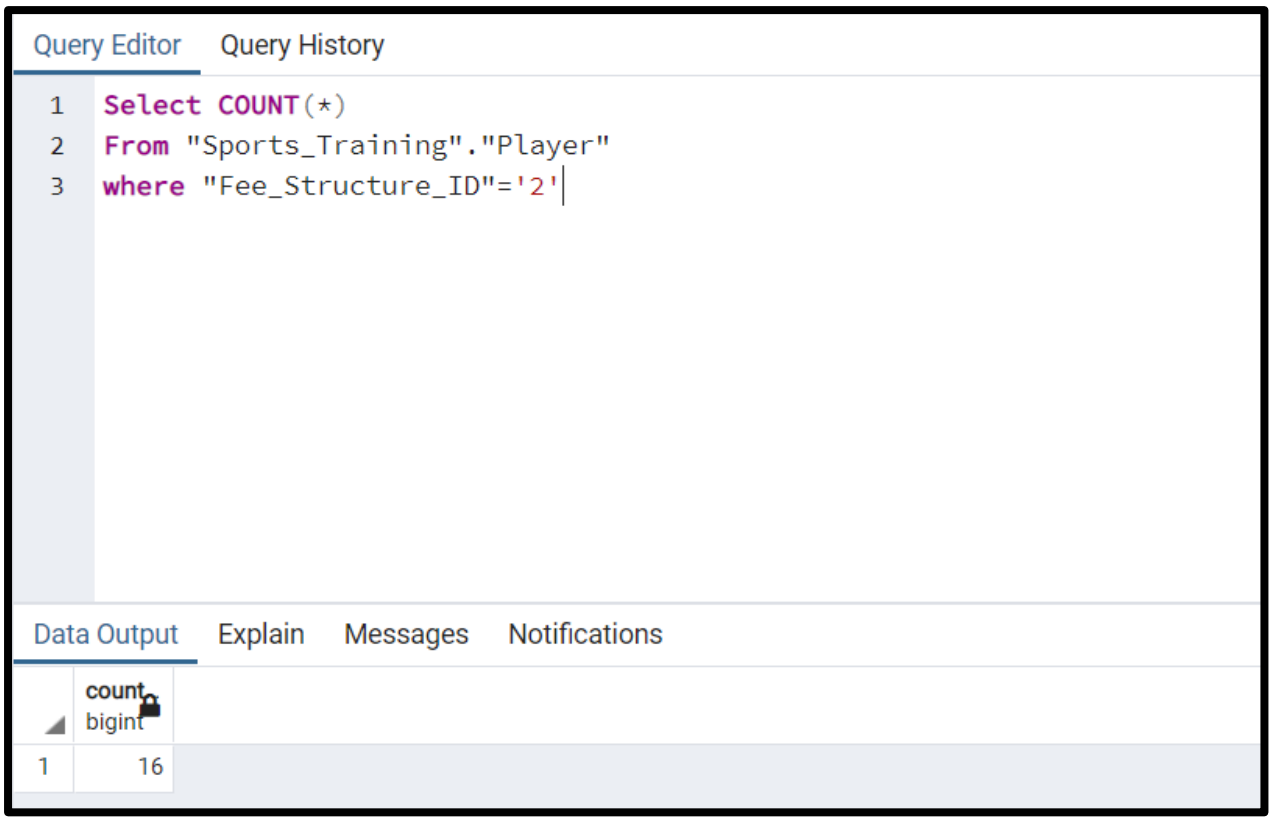
Trigger function for Calculating Age of Player:

```
1.  CREATE OR REPLACE FUNCTION "Sports_Training".age_1()
2.      RETURNS trigger
3.      LANGUAGE 'plpgsql'
4.      VOLATILE
5.      COST 100
6.  AS $BODY$
7.  Declare
8.  "diff" integer;
9.  Begin
10. Select  Current_Date- "Birth_Date"  into "diff" from "Sports_Training"."Player";
11. NEW."Age" := diff/365;
12. return new;
13. END
14. $BODY$;



15. CREATE TRIGGER age_1
16.     BEFORE INSERT OR UPDATE
17.     ON "Sports_Training"."Player"
18.     FOR EACH ROW
19.     EXECUTE FUNCTION "Sports_Training".age_1();
```

SQL Queries:

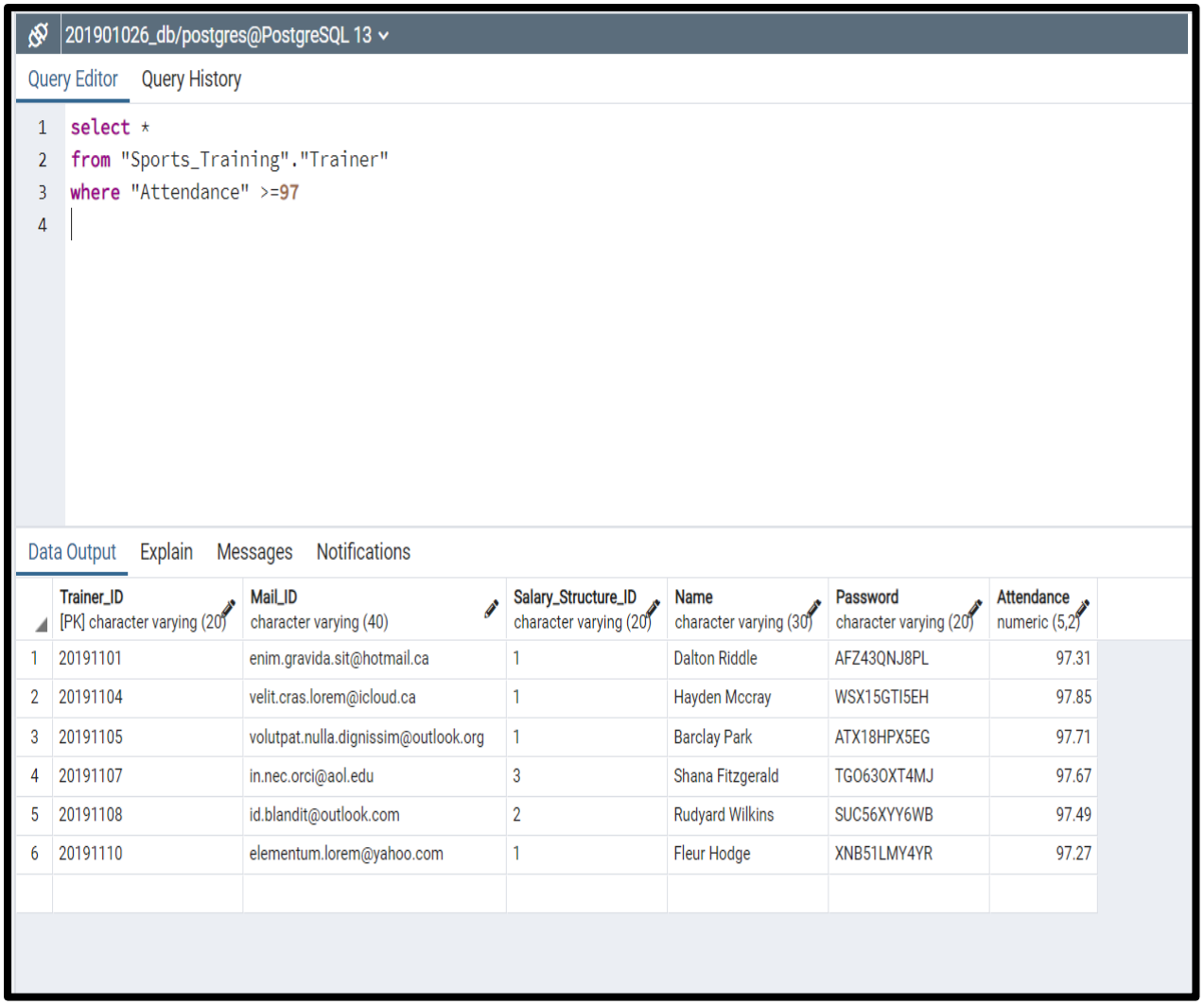
1. **Count all Students with Fee Structure id 2.**

```
1. Select COUNT(*)
2. From "Sports_Training"."Player"
3. where "Fee_Structure_ID"='2'
```



2. Find all the trainers with attendance greater than or equal to 97%.

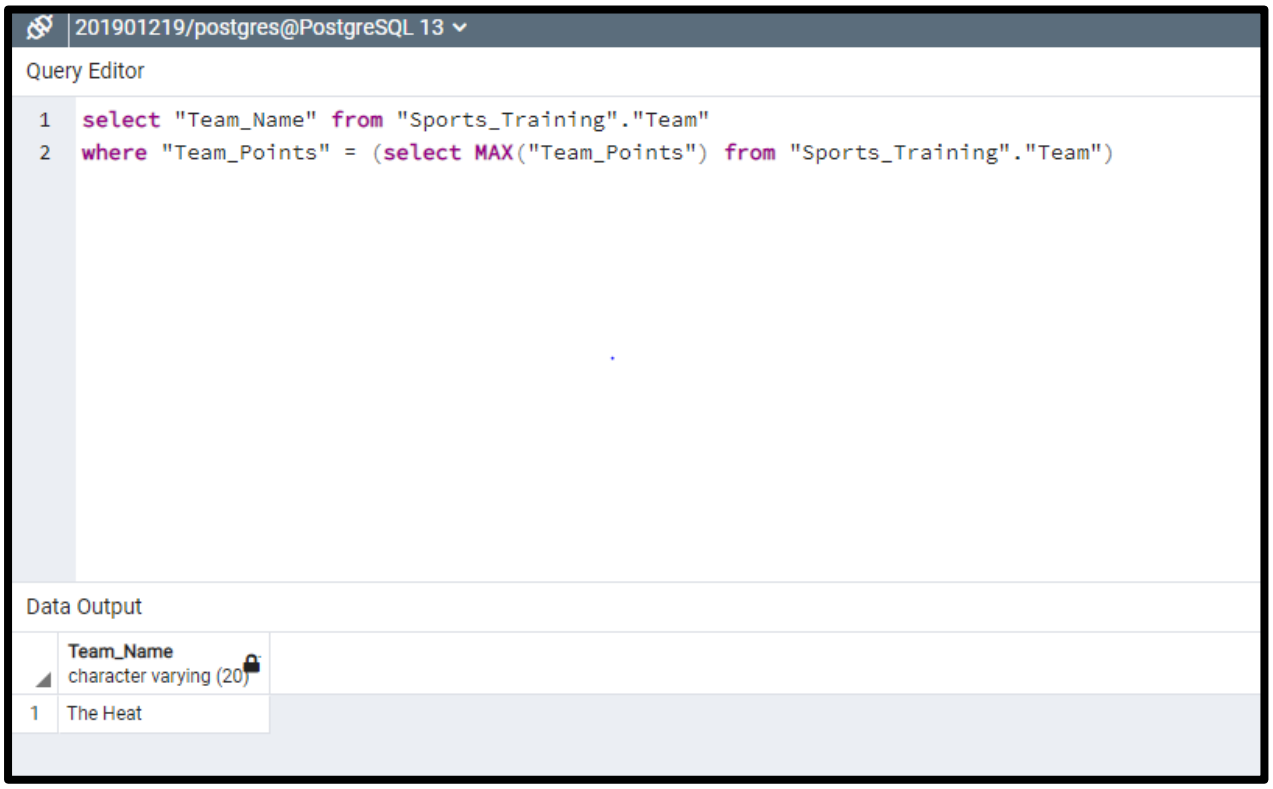
```
1. select *
2. from "Sports_Training"."Trainer"
3. where "Attendance" >=97
```



	Trainer_ID [PK] character varying (20)	Mail_ID character varying (40)	Salary_Structure_ID character varying (20)	Name character varying (30)	Password character varying (20)	Attendance numeric (5,2)
1	20191101	enim.gravida.sit@hotmail.ca	1	Dalton Riddle	AFZ43QNJ8PL	97.31
2	20191104	velit.cras.lorem@icloud.ca	1	Hayden Mccray	WSX15GTI5EH	97.85
3	20191105	volutpat.nulla.dignissim@outlook.org	1	Barclay Park	ATX18HPX5EG	97.71
4	20191107	in.nec.orci@aol.edu	3	Shana Fitzgerald	TGO63OXT4MJ	97.67
5	20191108	id.blandit@outlook.com	2	Rudyard Wilkins	SUC56XYY6WB	97.49
6	20191110	elementum.lorem@yahoo.com	1	Fleur Hodge	XNB51LMY4YR	97.27

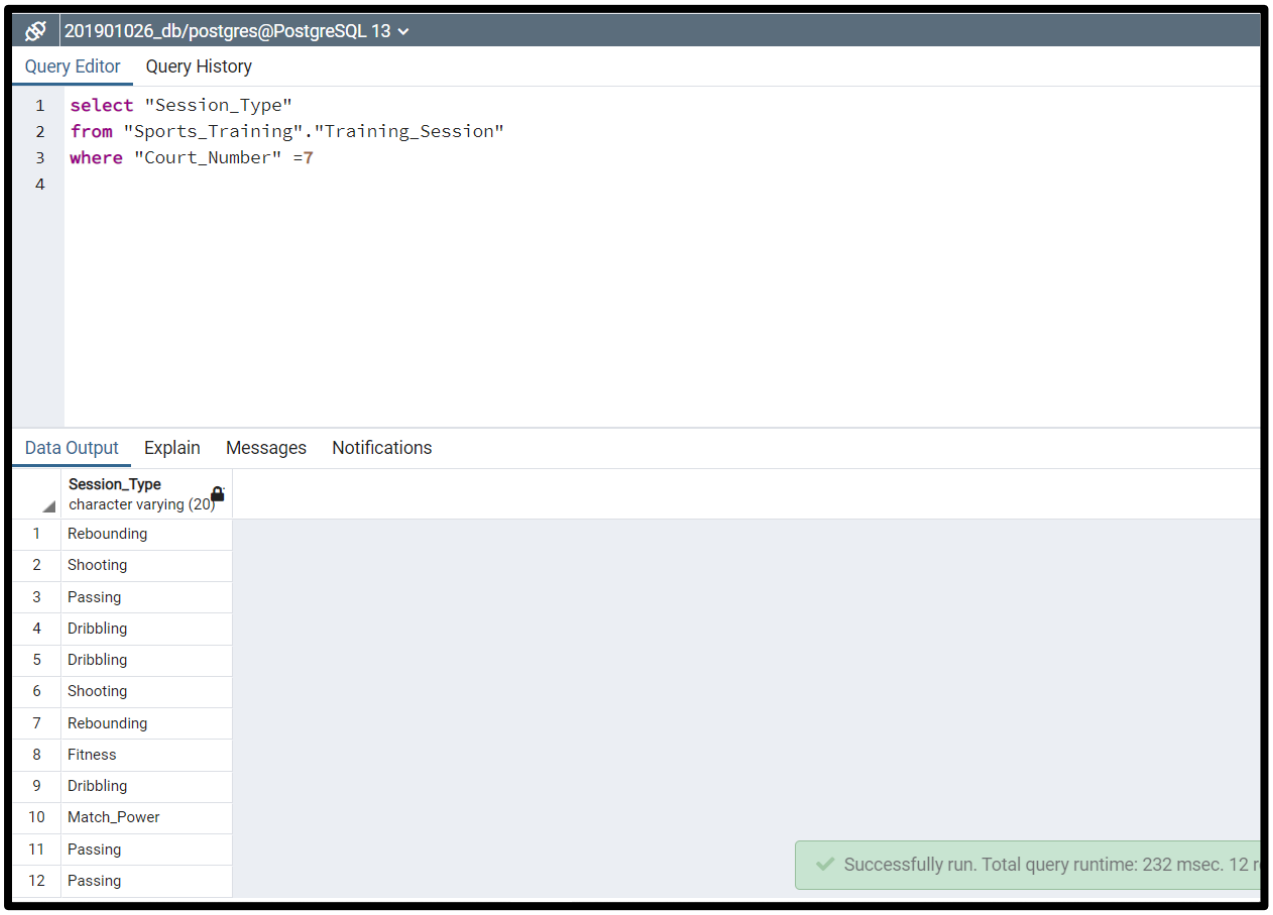
3. **Find the name of the team with the highest points.**

```
1. select "Team_Name" from "Sports_Training"."Team"
2. where "Team_Points" = (select MAX("Team_Points")
3. from "Sports_Training"."Team")
```



4. Find all types of sessions held at Court Number 7.

```
1. select "Session_Type"
2. from "Sports_Training"."Training_Session"
3. where "Court_Number" = 7
```



5. **Find top 10 players in dribbling skill based on grade card id = 1.**

```
1. select "Player_ID", "Dribbling"
2. from "Sports_Training"."Grade_Card"
3. where "Grade_Card_ID" = '1'
4. order by "Dribbling" DESC
5. limit 10
```

Basketball/postgres@PostgreSQL 13

Query Editor | Query History

```
1  select "Player_ID", "Dribbling"
2  from "Sports_Training"."Grade_Card"
3  where "Grade_Card_ID" = '1'
4  order by "Dribbling" DESC
5  limit 10
```

Data Output | Explain | Messages | Notifications

	Player_ID character varying (20)	Dribbling numeric (5,2)
1	201990	91.83
2	201958	89.80
3	201977	89.35
4	201940	89.21
5	201975	88.93
6	201957	88.86
7	201965	88.84
8	201953	88.43
9	201917	88.22
10	201964	88.03

6. Find the name and player status of players, whose height is greater than or equal to 165.

```
1. select ("Name" , "Player_Status")
2. from "Sports_Training"."Player"
3. where "Height" >=165
```

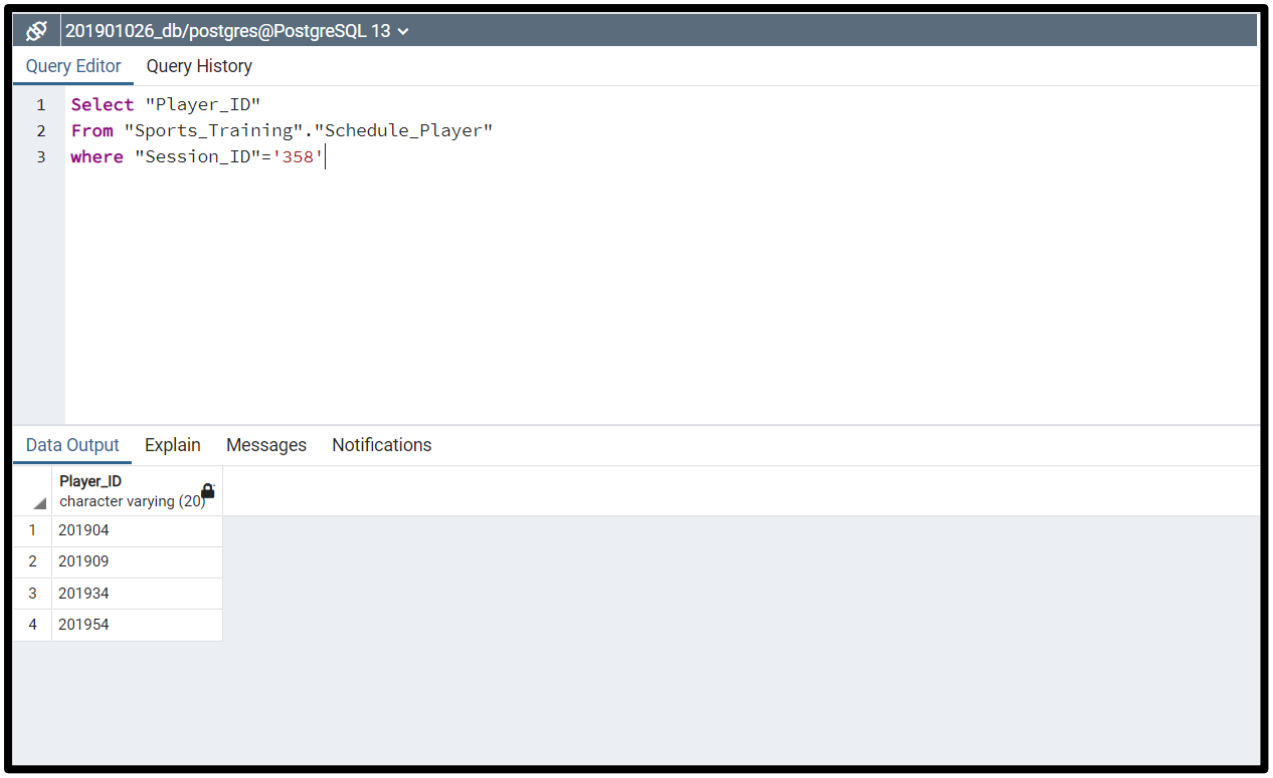
```
1  select ("Name" , "Player_Status")
2  from "Sports_Training"."Player"
3  where "Height" >=165
```

Data Output | Explain | Messages | Notifications

	row record
2	(Oleg Mccarthy ,Active)
3	("Madaline Moore",Active)
4	("Tanner Berry",Active)
5	("Hilel Parker",Active)
6	("Lisandra Walter",Active)
7	("Yvonne Hardy",Active)
8	("Adele Montgomery",Active)
9	("Emma Atkins",Active)
10	("Lila Stokes",Active)
11	("Unity Mcbride",Active)
12	("Acton Strickland",Active)
13	("Rhoda O'connor",Active)
14	("Britanni Hebert",Active)
15	("Sara Gibson",Active)

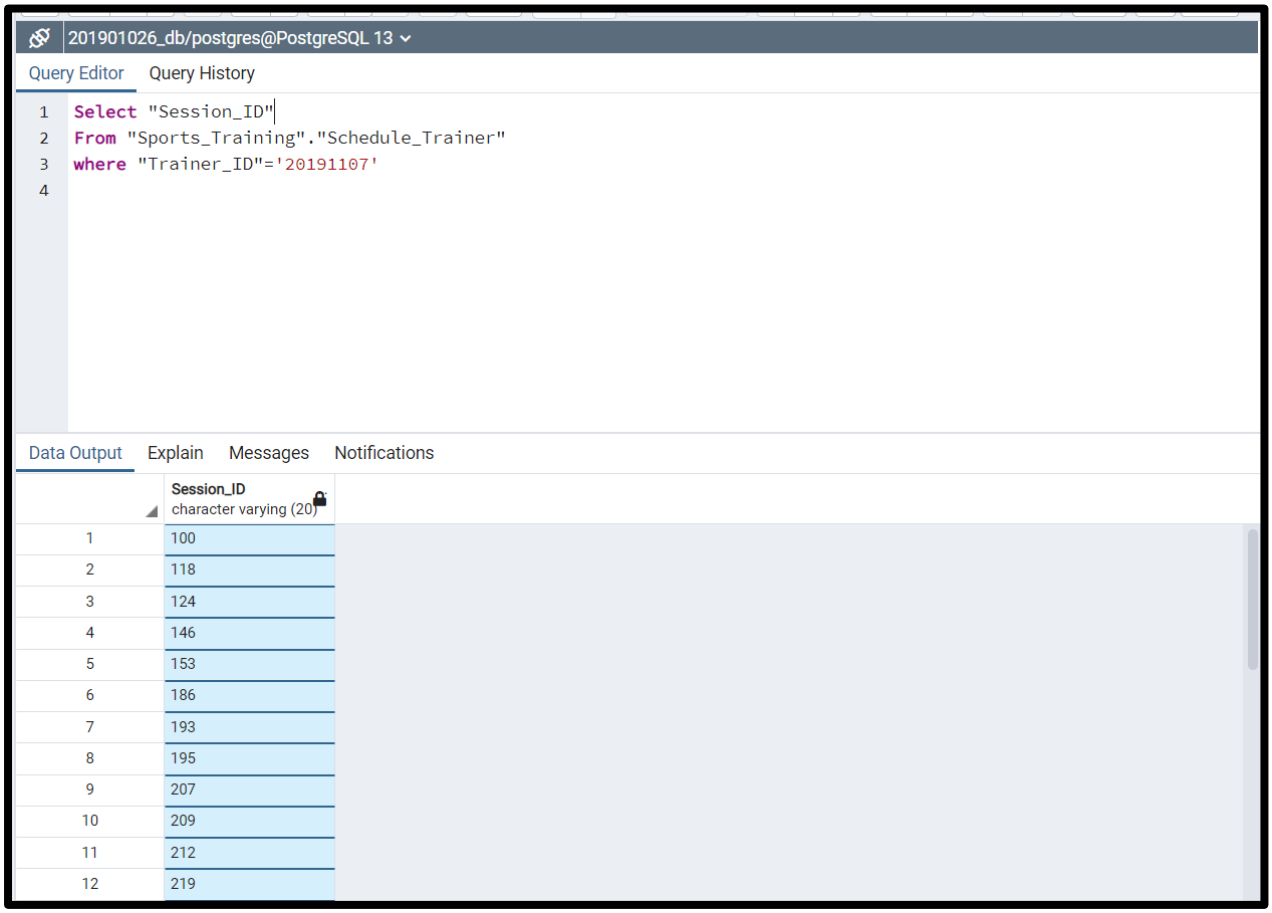
7. **List all Player_IDs who are taking part in the session with ID 358.**

```
1. Select "Player_ID"
2. From "Sports_Training"."Schedule_Player"
3. where "Session_ID"='358'
```



8. **List Session_ID of all the sessions taken by Trainer, whose id is 20191107.**

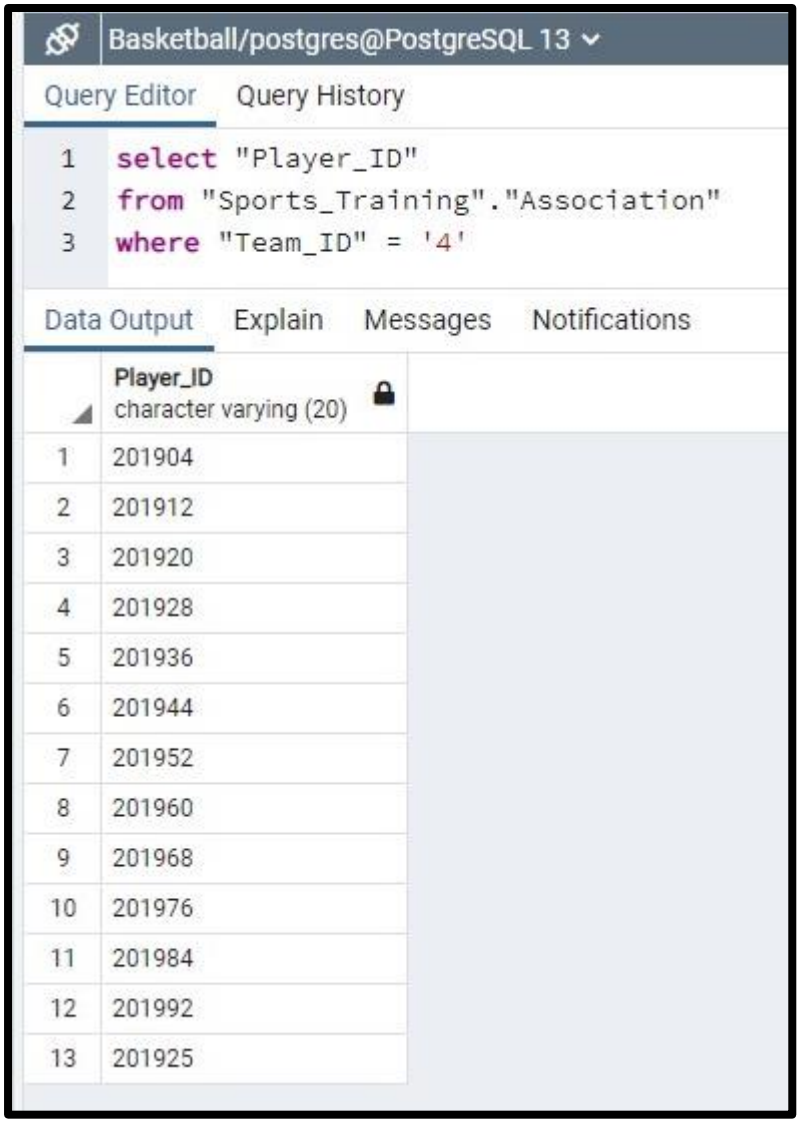
```
1. Select "Session_ID"
2. From "Sports_Training"."Schedule_Trainer"
3. where "Trainer_ID"='20191107'
```



Number of records = 30

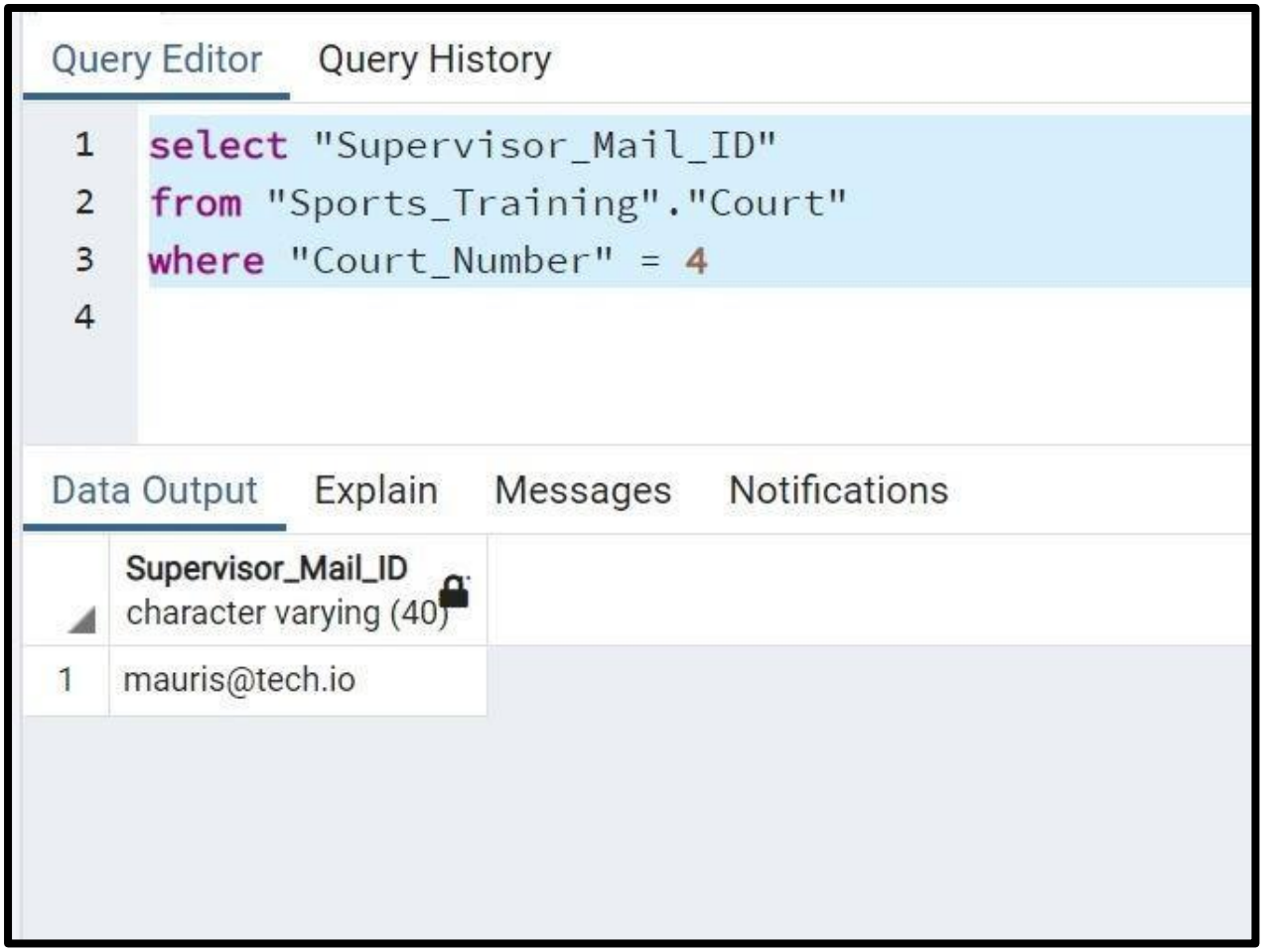
9. List all the players playing/played for Team with id 4.

```
1. select "Player_ID"
2. from "Sports_Training"."Association"
3. where "Team_ID" = '4'
```



10. Give the mail ID of the Supervisor of Court No.4.

```
1. select "Supervisor_Mail_ID"
2. from "Sports_Training"."Court"
3. where "Court_Number" = 4
```



11. List all the Session_ID of matches played by Team with id 6.

```
1. select "Session_ID"
2. from "Sports_Training"."Team_Match"
3. where "Team_ID" = '6'
```

Query Editor | Query History

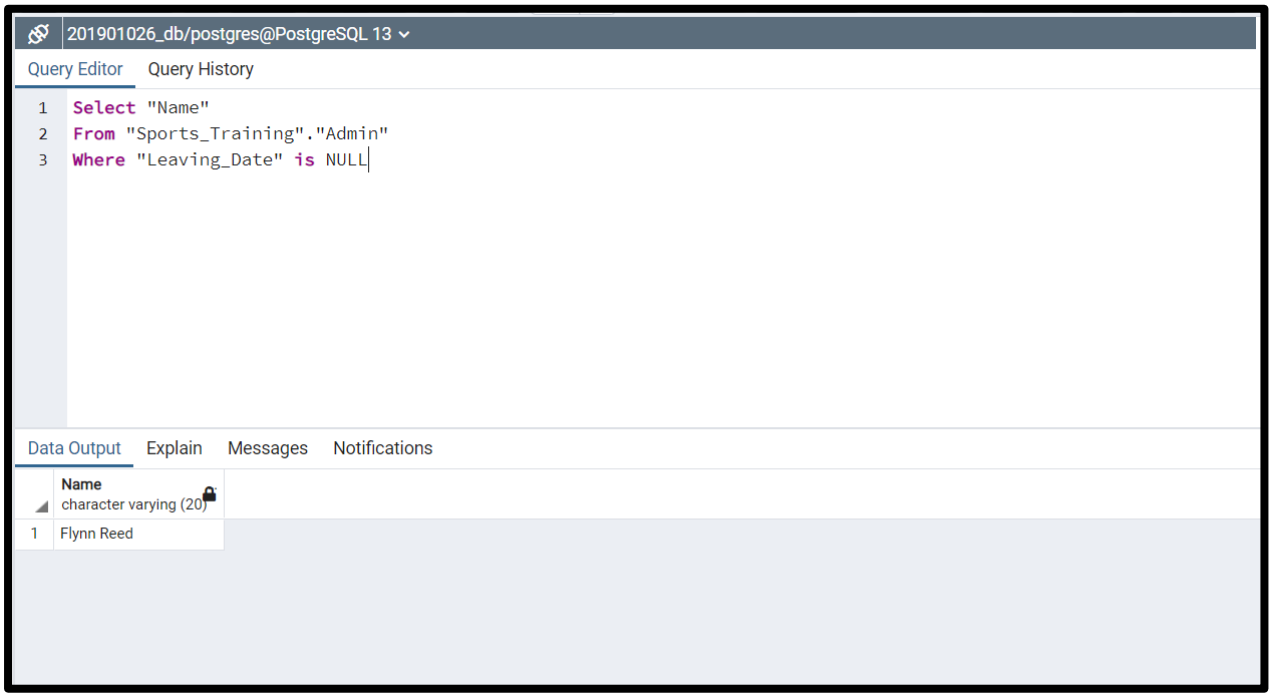
```
1  select "Session_ID"
2  from "Sports_Training"."Team_Match"
3  where "Team_ID" = '6'
4
```

Data Output | Explain | Messages | Notifications

	Session_ID character varying (20)
1	142
2	324

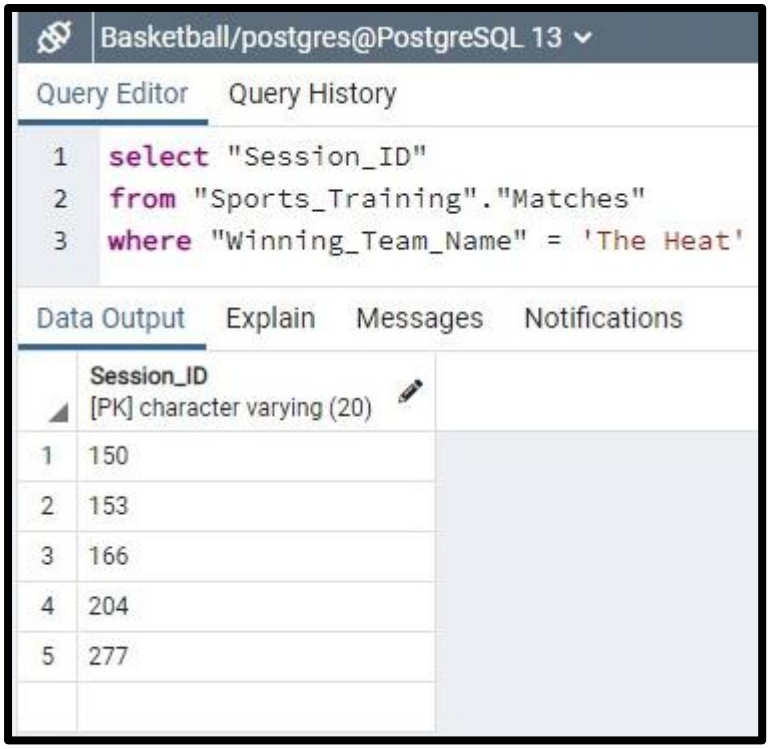
12. Who is currently working as an Admin for the Training Management System?

```
1. Select "Name"
2. From "Sports_Training"."Admin"
3. Where "Leaving_Date" is NULL
```



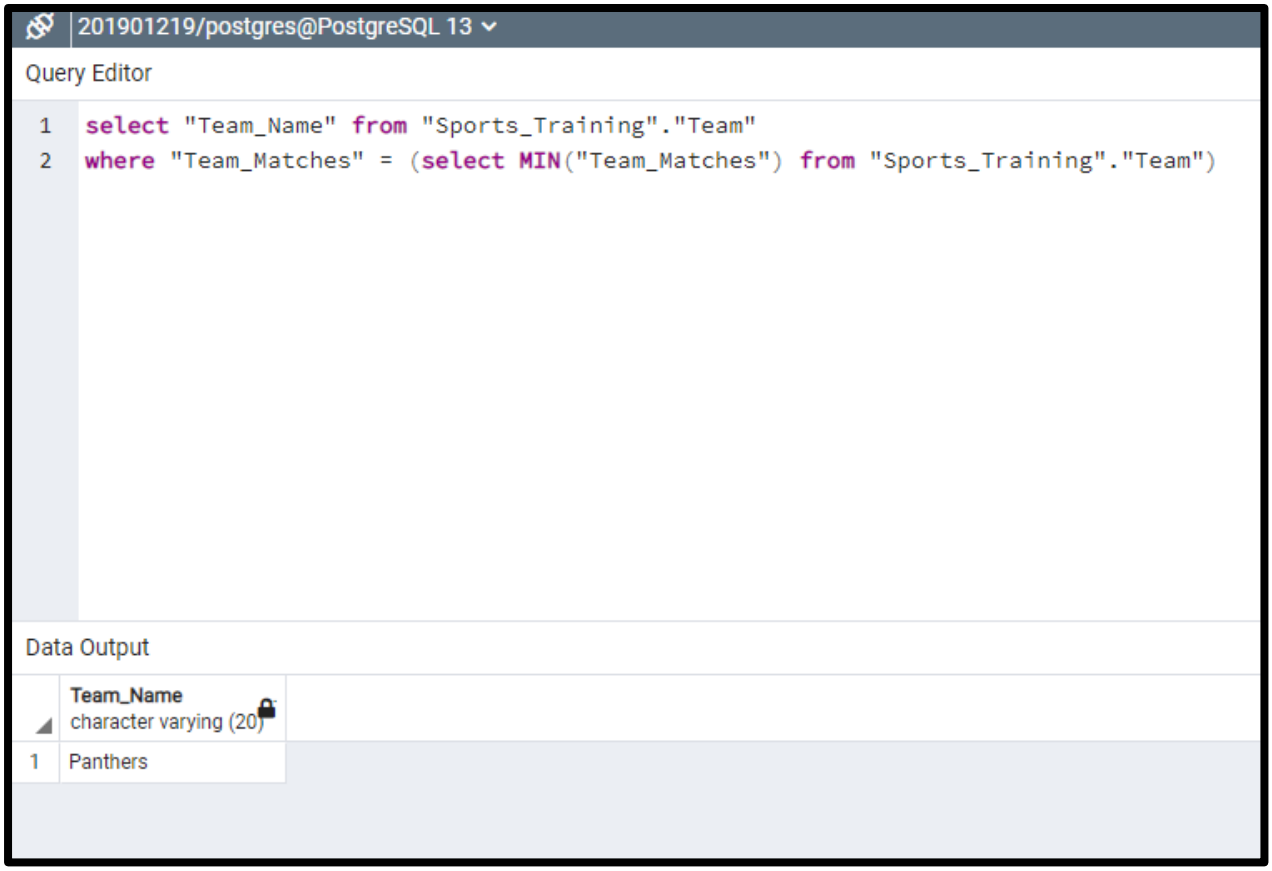
13. List the Session_ID of all the matches won by Team, “The Heat”.

```
1. select "Session_ID"
2. from "Sports_Training"."Matches"
3. where "Winning_Team_Name" = 'The Heat'
```



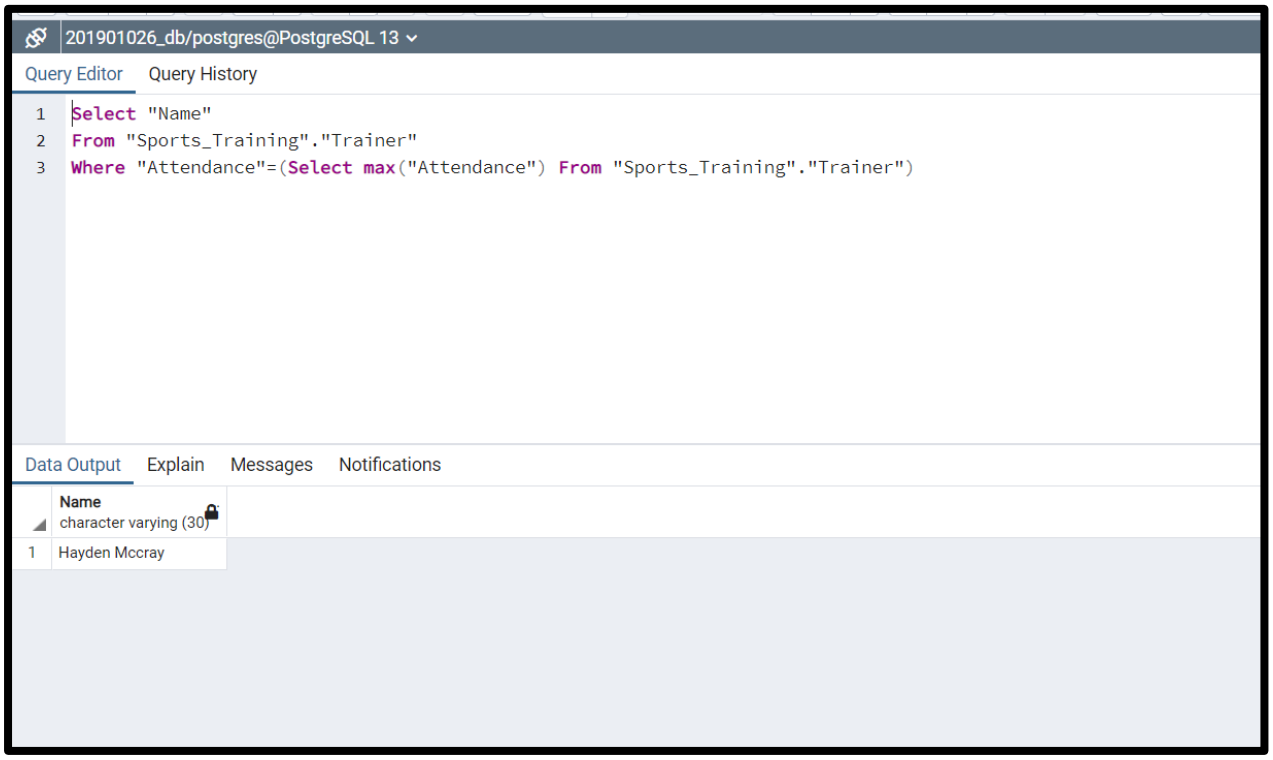
14. Find the team which overall played lowest matches among all the teams.

```
1. select "Team_Name" from "Sports_Training"."Team"
2. where "Team_Matches" = (select MIN("Team_Matches") from "Sports_Training"."Team")
```



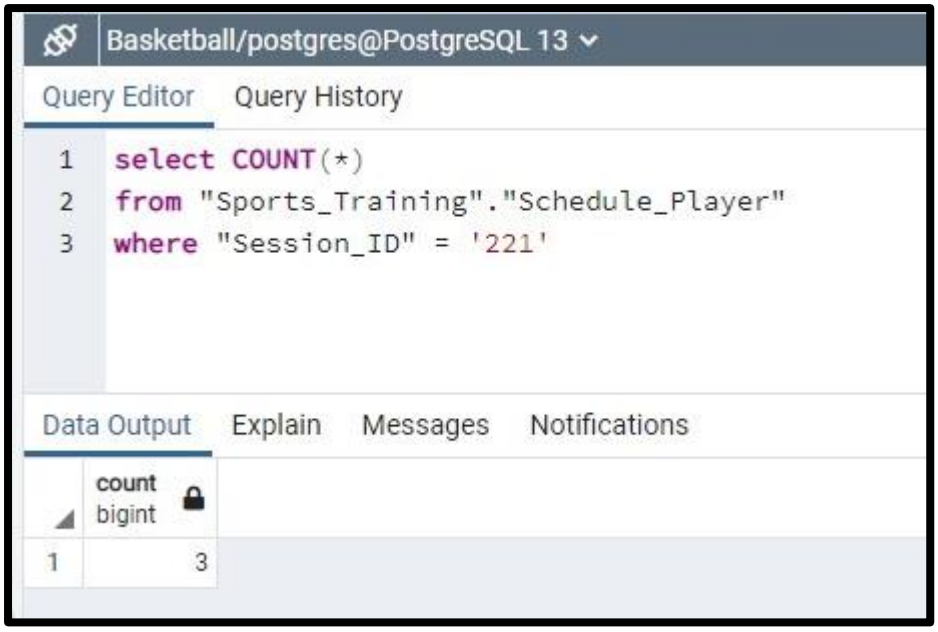
15. Which trainer has the highest attendance among all trainers?

```
1. Select "Name"
2. From "Sports_Training"."Trainer"
3. Where "Attendance"=(Select max("Attendance") From "Sports_Training"."Trainer")
```



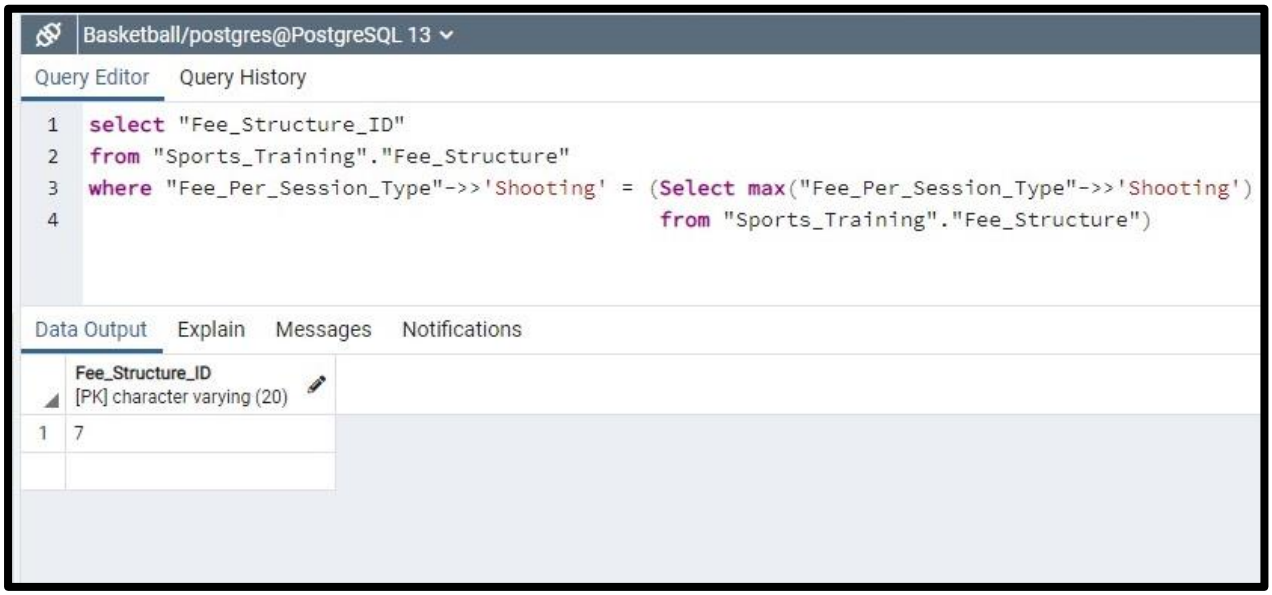
16. Count the number of Players present in Session ID, 221.

```
1. select COUNT(*)
2. from "Sports_Training"."Schedule_Player"
3. where "Session_ID" = '221'
```



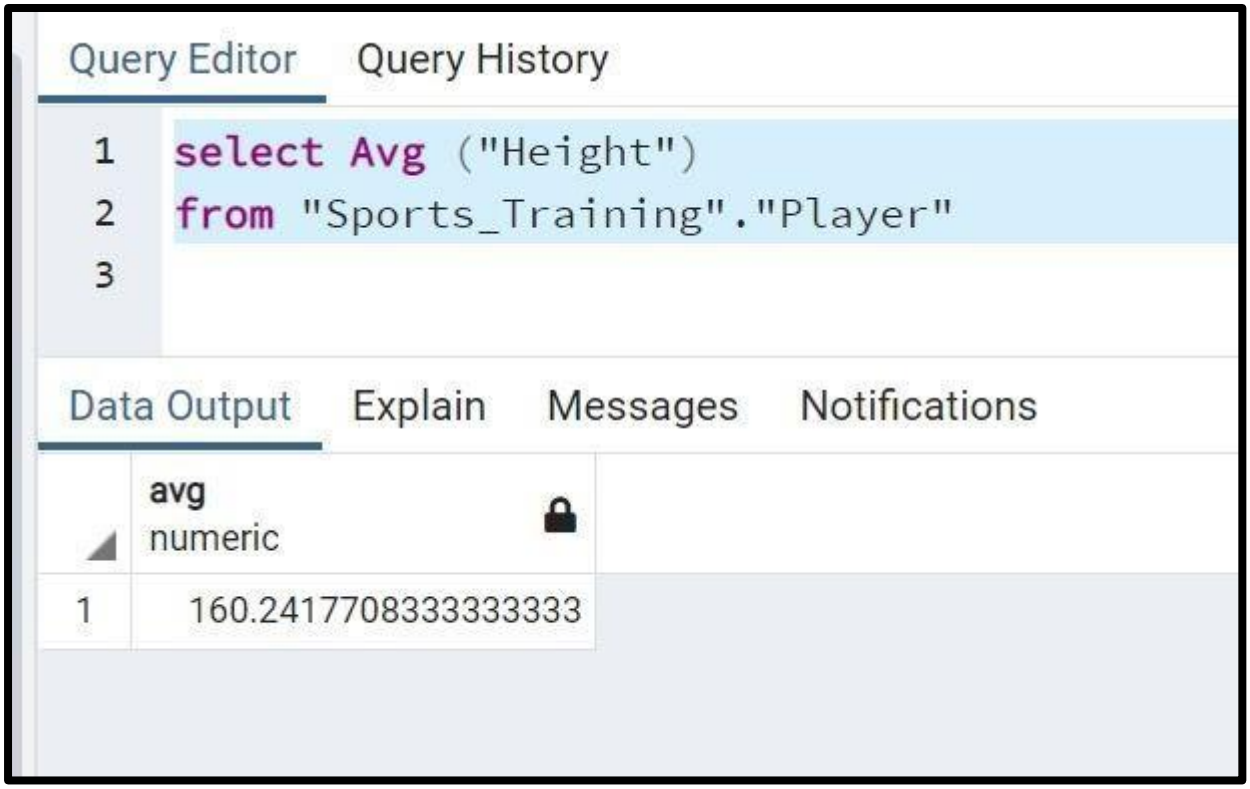
17. List the Fee_Structure_ID of Fee_Structure which has the highest fee for Shooting.

```
1. select "Fee_Structure_ID"
2. from "Sports_Training"."Fee_Structure"
3. where "Fee_Per_Session_Type"->>'Shooting' = (Select max("Fee_Per_Session_Type"->>'Shooting')
   rom "Sports_Training"."Fee_Structure")
```



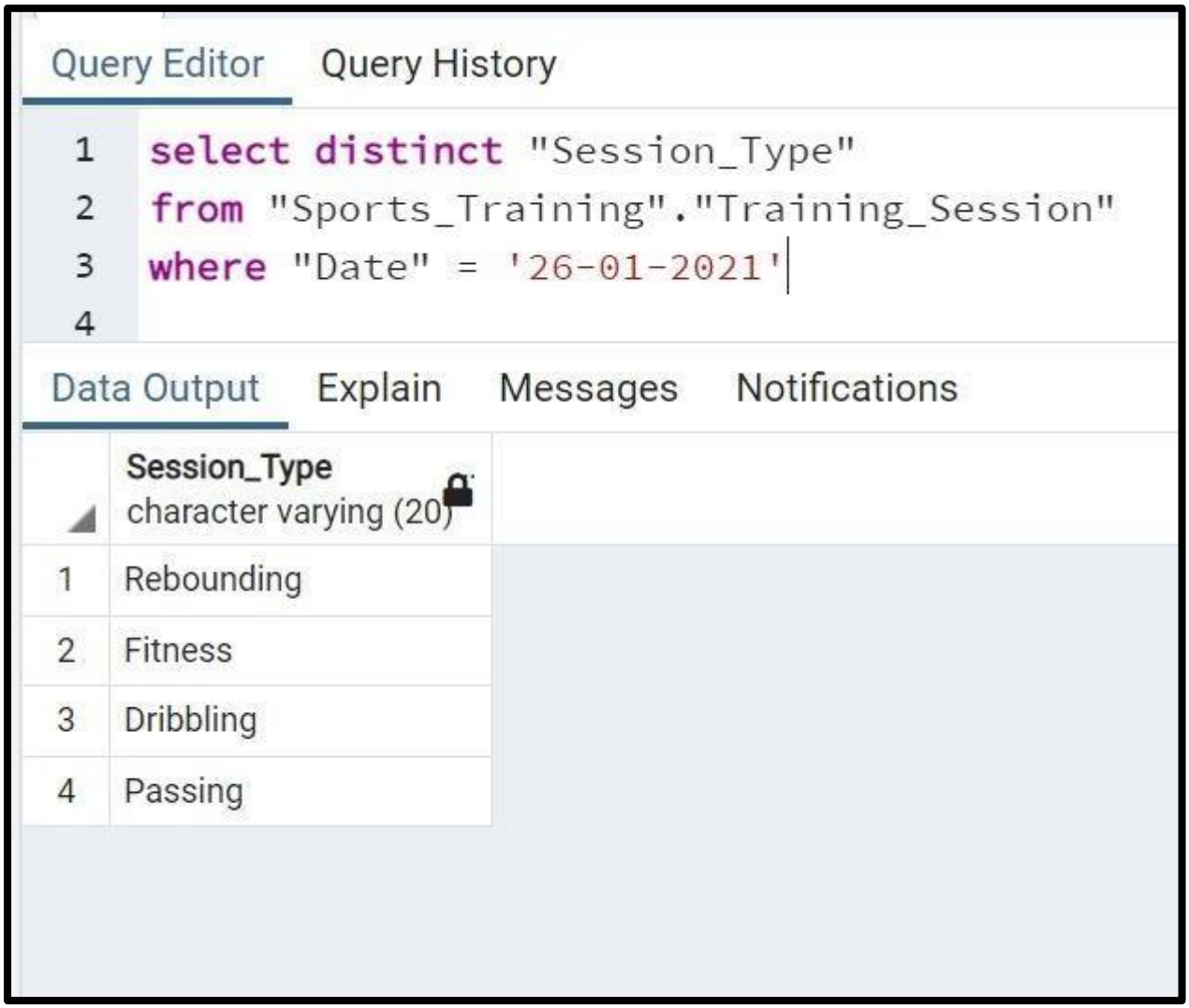
18. Find the average height of players enrolled in the academy.

```
1. select Avg ("Height")
2. from "Sports_Training"."Player"
```



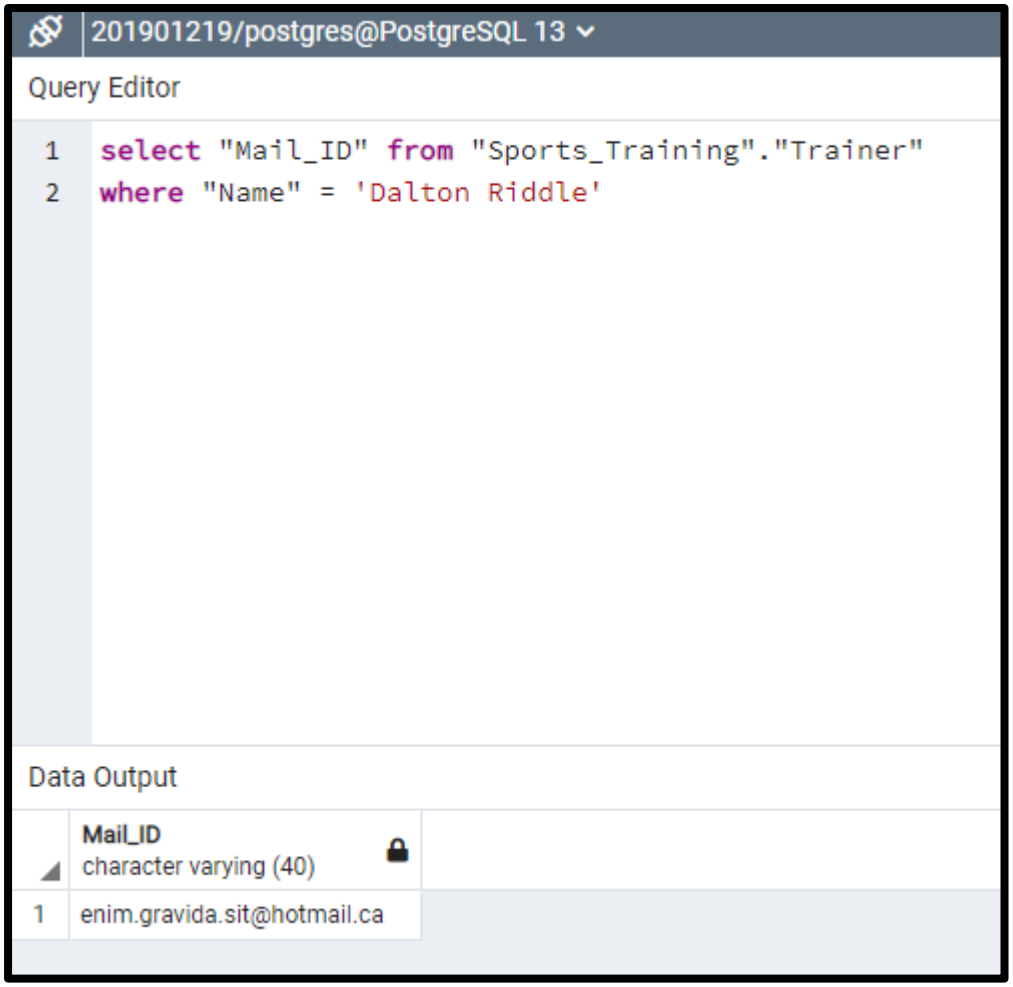
19. List all the session-types , which are scheduled on 26th January, 2021.

```
1. select distinct "Session_Type"
2. from "Sports_Training"."Training_Session"
3. where "Date" = '26-01-2021'
```



20. Find the mail id of a trainer whose name is Dalton Riddle.

```
1. select "Mail_ID" from "Sports_Training"."Trainer"
2. where "Name" = 'Dalton Riddle'
```



Complex Queries:

21. List all player's names who played more than 50 matches for one team.

```
1. select "Name"
2. from "Sports_Training"."Player"
3. natural join "Sports_Training"."Association"
4. where "Matches_Played" >50
```

Query Editor | Query History

```
1  select "Name"
2  from "Sports_Training"."Player"
3  natural join "Sports_Training"."Association"
4  where "Matches_Played" >50
```

Data Output | Explain | Messages | Notifications

	Name character varying (20)
1	Lareina Green
2	Kibo Allen
3	Madaline Moore
4	Kenyon Rose
5	Kirsten Hartman
6	Lisandra Walter
7	Harding Barrera
8	Cole Hunter
9	Doris Avila
10	Emma Cox

Number of records = 37

22. List all player's names who have match power more than 80 in any grade report.

```
1. select distinct "Name"
2. from "Sports_Training"."Player"
3. natural join "Sports_Training"."Grade_Card"
4. where "Match_Power" >80
```

Query Editor | Query History

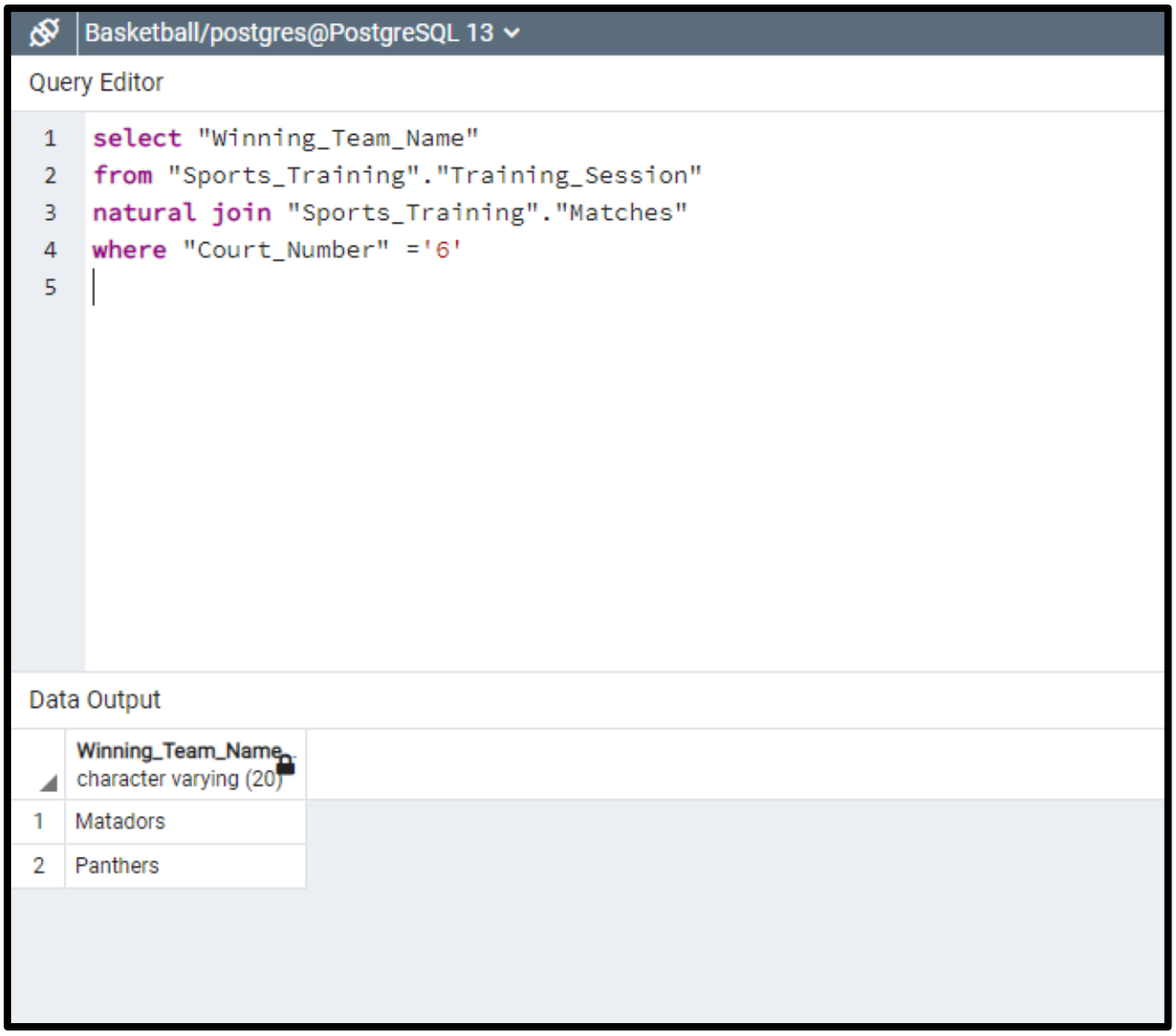
```
1  select distinct "Name"
2  from "Sports_Training"."Player"
3  natural join "Sports_Training"."Grade_Card"
4  where "Match_Power" >80
```

Data Output | Explain | Messages | Notifications

	Name character varying (20)
1	Adele Montgomery
2	Jordan Wallace
3	Keith Guthrie
4	Kyle Holmes
5	Madaline Moore

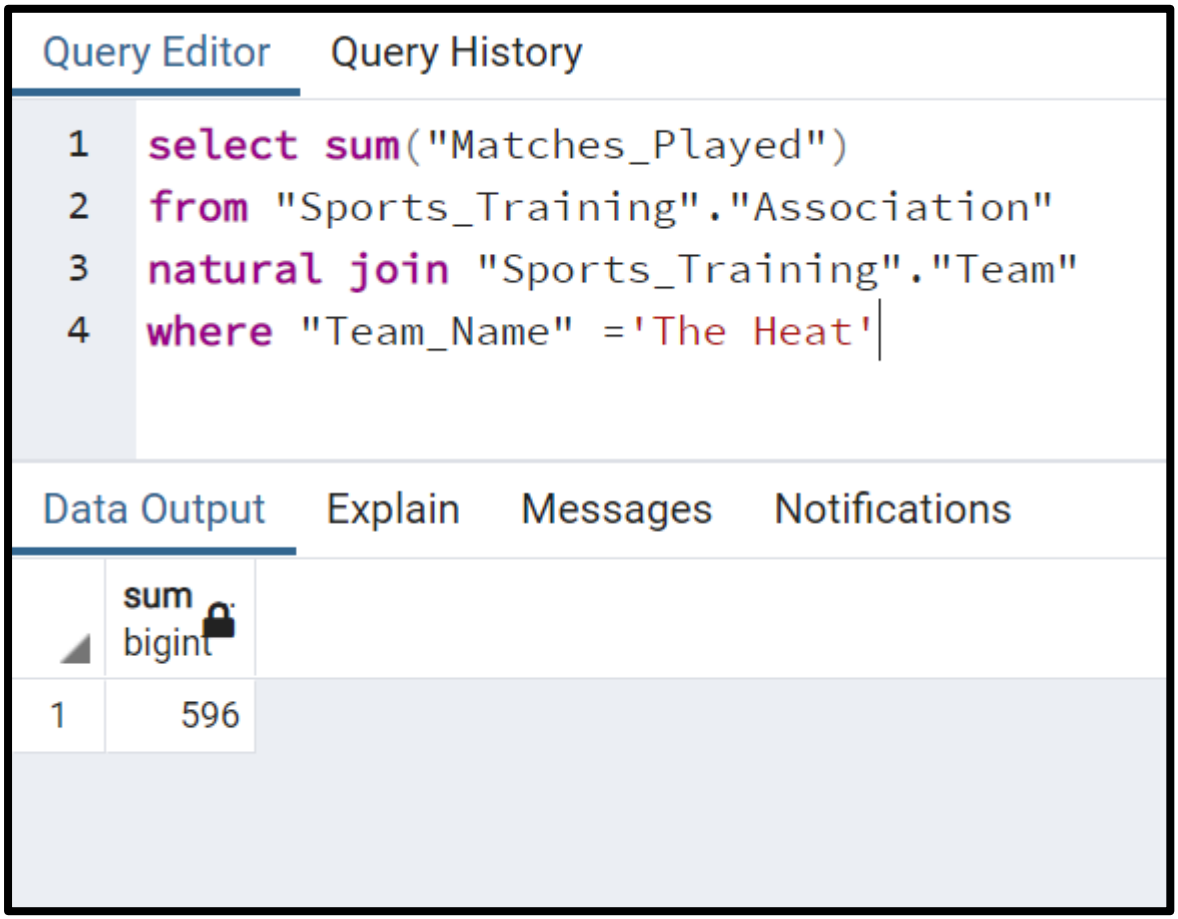
23. Find name of all Teams who had won match in Court number 6.

```
1. select "Winning_Team_Name"
2. from "Sports_Training"."Training_Session"
3. natural join "Sports_Training"."Matches"
4. where "Court_Number" ='6'
```



24. Find the sum of matches played by players of the team “The Heat”.

```
1. select sum("Matches_Played")
2. from "Sports_Training"."Association"
3. natural join "Sports_Training"."Team"
4. where "Team_Name" ='The Heat'
```



25.List all player's names and mobile numbers who have attendance less than 88.

```
1. select "Name","Mobile_Number1","Mobile_Number2"
2. from "Sports_Training"."Contact_Information"
3. natural join "Sports_Training"."Player"
4. where "Attendance" <88
```

Query Editor | Query History

```
1  select "Name","Mobile_Number1","Mobile_Number2"
2  from "Sports_Training"."Contact_Information"
3  natural join "Sports_Training"."Player"
4  where "Attendance" <88
```

Data Output | Explain | Messages | Notifications

	Name character varying (20)	Mobile_Number1 character varying (20)	Mobile_Number2 character varying (20)
1	Aurelia Perkins	(151) 682-2572	1-776-914-5971
2	Regina Hatfield	1-182-541-3634	1-467-756-1427
3	Coby Pickett	1-218-268-5454	1-416-155-2558
4	Madaline Moore	(966) 282-1459	1-556-436-6814
5	Kirsten Hartman	1-860-857-0751	(352) 536-3853
6	Ahmed Delgado	1-670-767-7884	1-358-732-3536
7	Emma Cox	(334) 726-2256	(888) 412-3221
8	Raven Ballard	(217) 327-4856	1-398-393-5762
9	Troy Castaneda	1-733-455-1844	1-946-554-6233
10	Kaye Crane	(363) 900-1707	1-117-628-2802
11	Kyla Thompson	1-168-348-0278	(555) 356-2503
12	Tucker Brennan	(264) 561-7347	(185) 835-8373
13	Martin Williams	(851) 358-7541	1-574-793-0515

Number of records = 16

26. List all players whose joining year is 2019 for any team.

```
1. select "Name"
2. from "Sports_Training"."Player"
3. natural join "Sports_Training"."Association"
4. where "Joining_Year" =2019
```

Query Editor | Query History

```
1  select "Name"
2  from "Sports_Training"."Player"
3  natural join "Sports_Training"."Association"
4  where "Joining_Year" =2019
```

Data Output | Explain | Messages | Notifications

	Name character varying (20)
1	Madaline Moore
2	Tanner Berry
3	Malik Weaver
4	Mollie Sears
5	Raven Ballard
6	Armand Pace
7	Kaye Crane
8	Adele Hopper
9	Shelby Hester
10	Luke Hancock
11	Lillith Stevenson
12	Zachery Richard

27. Find all player's names who played the highest matches.

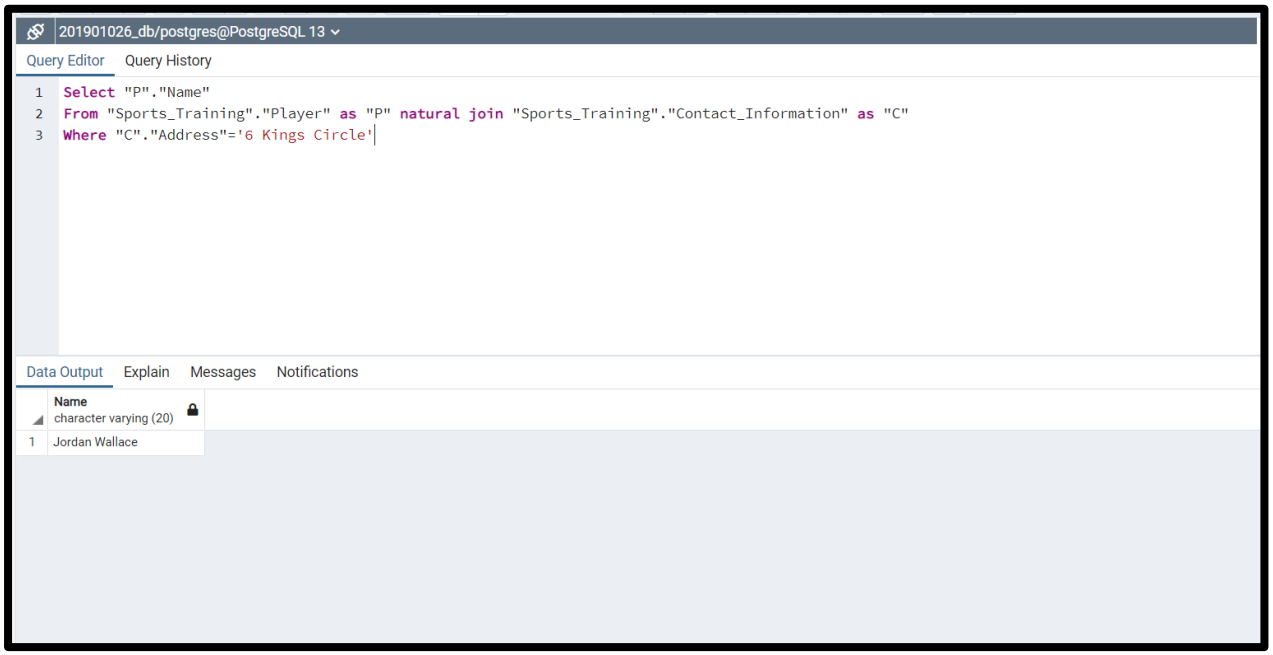
```
1. select distinct "Name"
2. from "Sports_Training"."Player"
3. natural join "Sports_Training"."Association"
4. where "Matches_Played" =(select max("Matches_Played")
5. from "Sports_Training"."Association")
```

28. List the average match power of the team: Shockers, based on 2nd grade report. (considering current players only)

```
1. select avg("Match_Power")
2. from "Sports_Training"."Team" natural join "Sports_Training"."Association" natural join
   "Sports_Training"."Grade_Card"
3. where "Team_Name" = 'Shockers' and "Grade_Card_ID" = '2' and "Status" = 'Current'
```

29 Give the name of the player who resides at "6 Kings Circle"?

```
1. Select "P"."Name"
2. From "Sports_Training"."Player" as "P" natural join "Sports_Training"."Contact_Information" as "C"
3. Where "C"."Address"='6 Kings Circle'
```

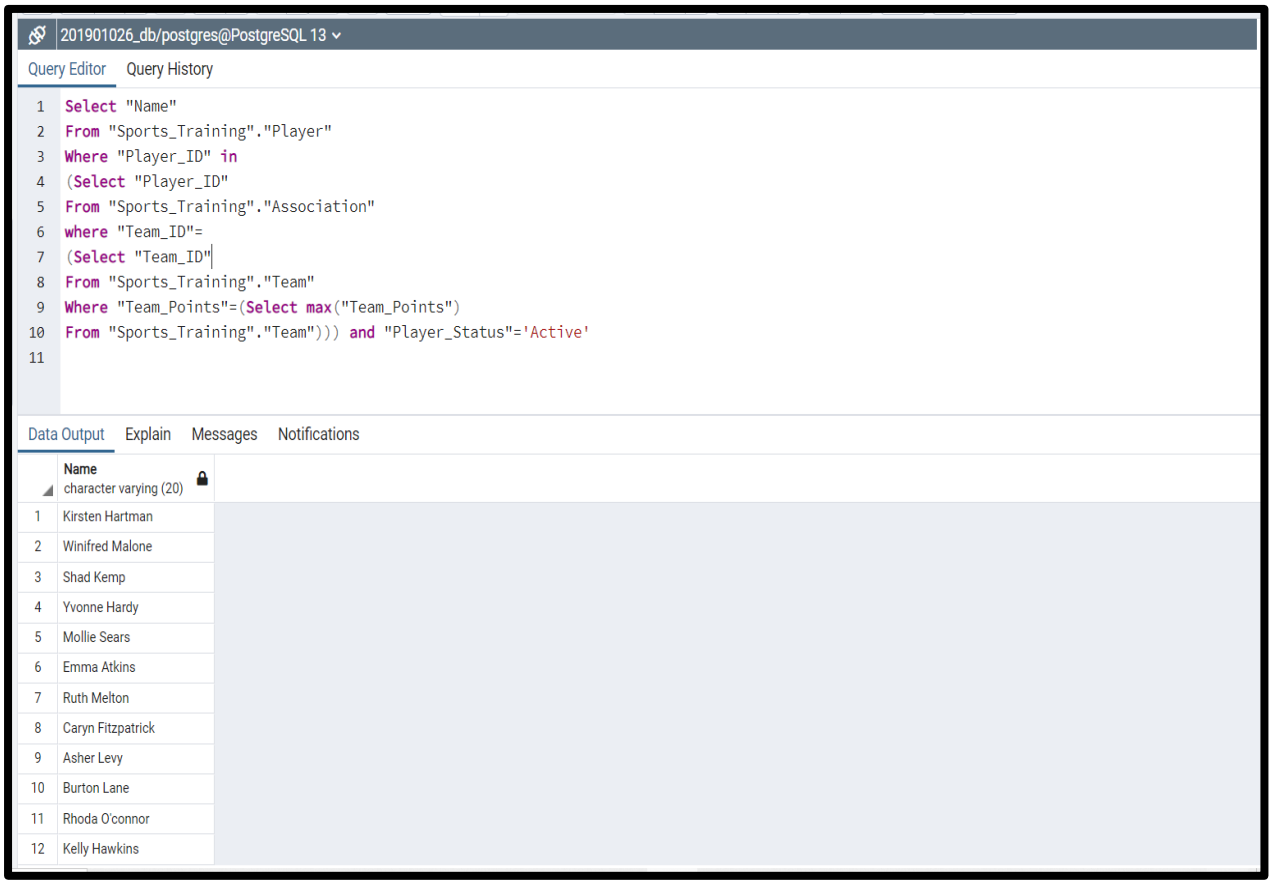


30. Calculate the Total Number of Passing Training Sessions held at Court 14.

```
1. select COUNT(*)
2. from "Sports_Training"."Court" natural join "Sports_Training"."Training_Session"
3. where "Court_Number" = 14 and "Session_Type" = 'Passing'
```

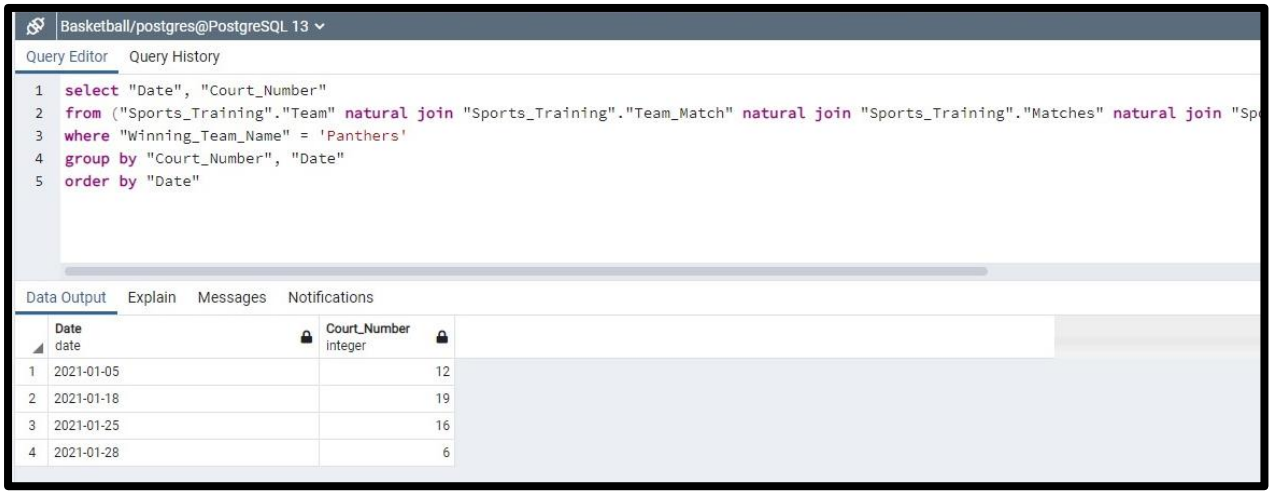
31. List all the player names of team who has scored highest points among other teams.

```
1.  Select "Name"
2.  From "Sports_Training"."Player"
3.  Where "Player_ID" in
4.  (Select "Player_ID"
5.  From "Sports_Training"."Association"
6.  where "Team_ID"=
7.  (Select "Team_ID"
8.  From "Sports_Training"."Team"
9.  Where "Team_Points"=(Select max("Team_Points")
10. From "Sports_Training"."Team"))) and "Player_Status"='Active'
```



32. List Date and Court Number for the matches won by team “Panthers”.

```
1. select "Date", "Court_Number"
2. from ("Sports_Training"."Team" natural join "Sports_Training"."Team_Match" natural join
   "Sports_Training"."Matches" natural join "Sports_Training"."Training_Session") as rec
3. where "Winning_Team_Name" = 'Panthers'
4. group by "Court_Number", "Date"
5. order by "Date"
```



33. Who has scored highest in Fitness in any grade report?

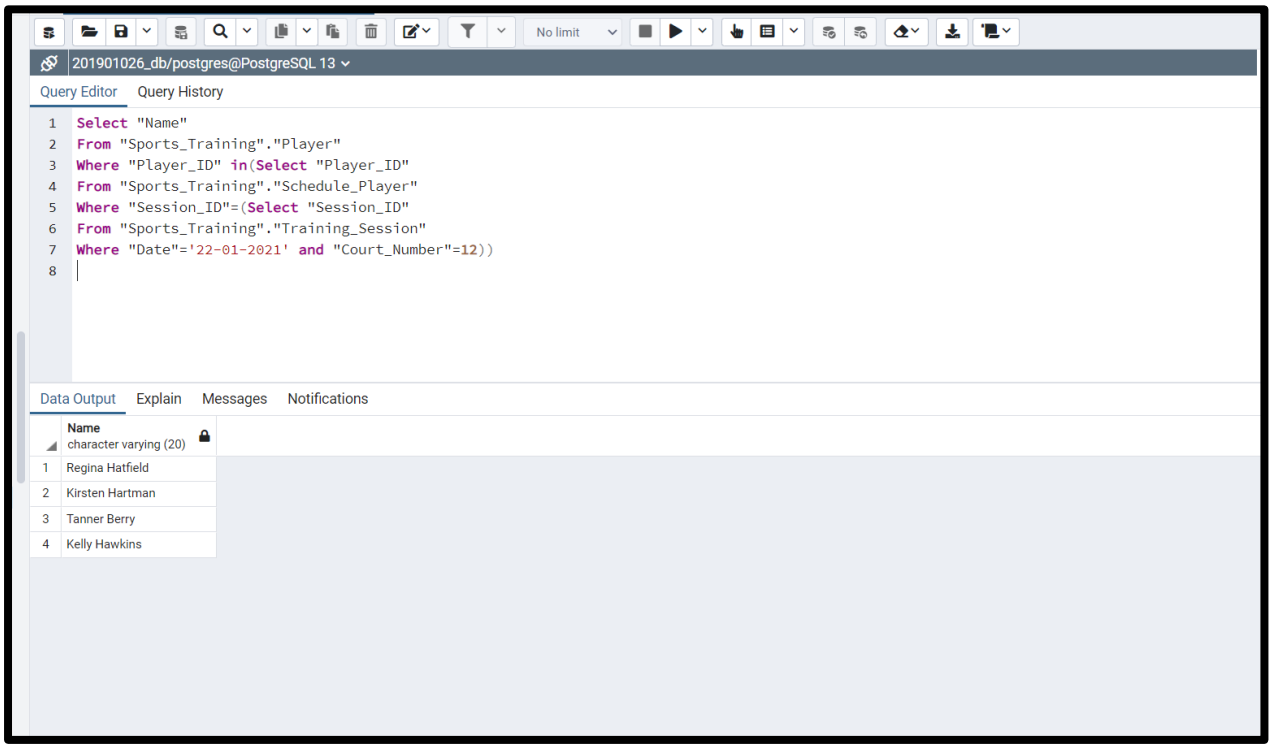
```
1. Select "Name"
2. From "Sports_Training"."Player"
3. Where "Player_ID"=(Select "Player_ID"
4. From "Sports_Training"."Grade_Card"
5. Where "Fitness"=(Select max("Fitness")
6. From "Sports_Training"."Grade_Card"))
```

34. Calculate number of sessions by Session Type attended by player whose id is 201965.

```
1. select "Session_Type", COUNT(*)
2. from "Sports_Training"."Player" natural join "Sports_Training"."Schedule_Player" natural join
   "Sports_Training"."Training_Session"
3. where "Player_ID" = '201965'
4. group by "Session_Type"
```

35. List all the player's names who played on 22nd January, 2001 at court No. 12.

```
1. Select "Name"
2. From "Sports_Training"."Player"
3. Where "Player_ID" in(Select "Player_ID"
4. From "Sports_Training"."Schedule_Player"
5. Where "Session_ID"=(Select "Session_ID"
6. From "Sports_Training"."Training_Session"
7. Where "Date"='22-01-2021' and "Court_Number"=12))
```

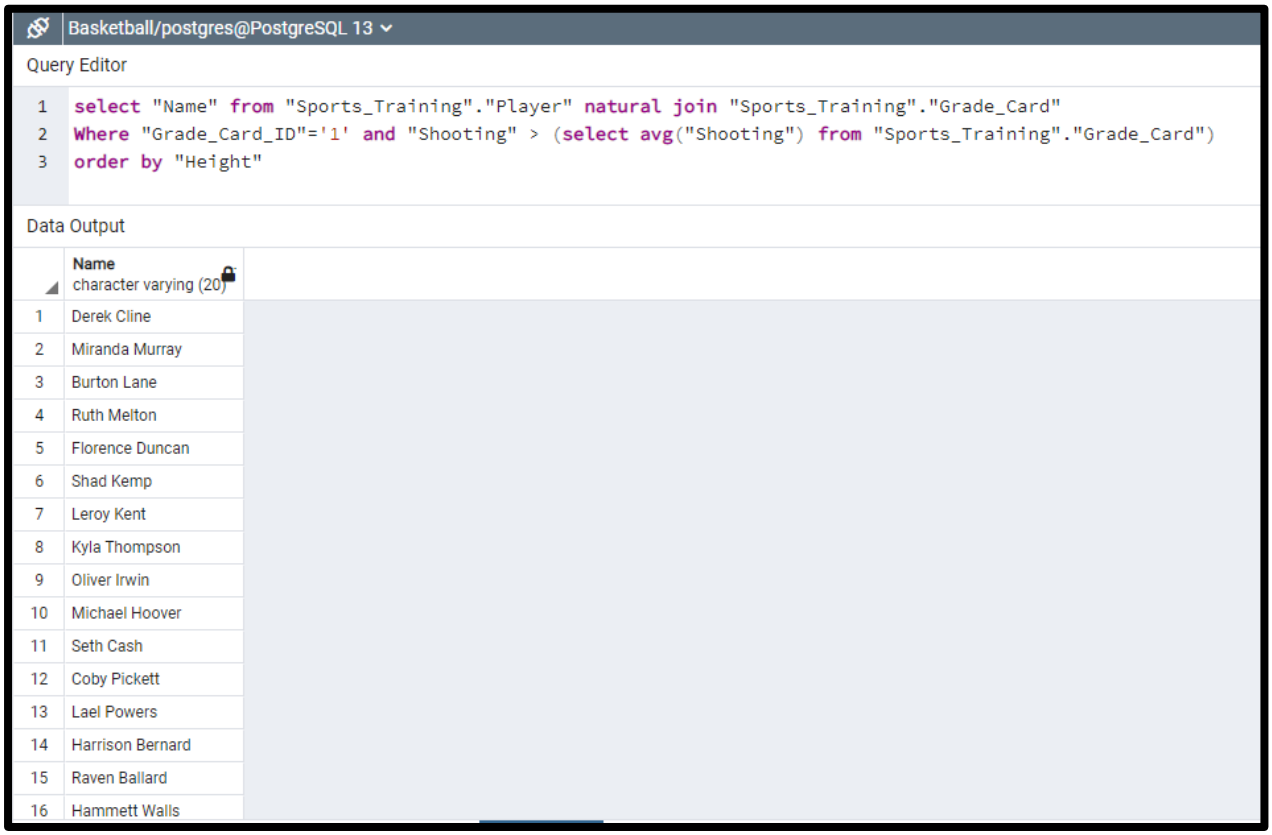


36. List the Date, Time and Court Number for Defence Sessions taken by Trainer whose id is 20191107.

```
1. select "Date", "Start_Time", "Court_Number"
2. from "Sports_Training"."Trainer" natural join "Sports_Training"."Schedule_Trainer" natural join
   "Sports_Training"."Training_Session"
3. where "Trainer_ID" = '20191107' and "Session_Type" = 'Defence'
4. order by "Date"
```

37. Find name of the students whose shooting is greater than average in Grade_Card_ID =1. And order it by height.

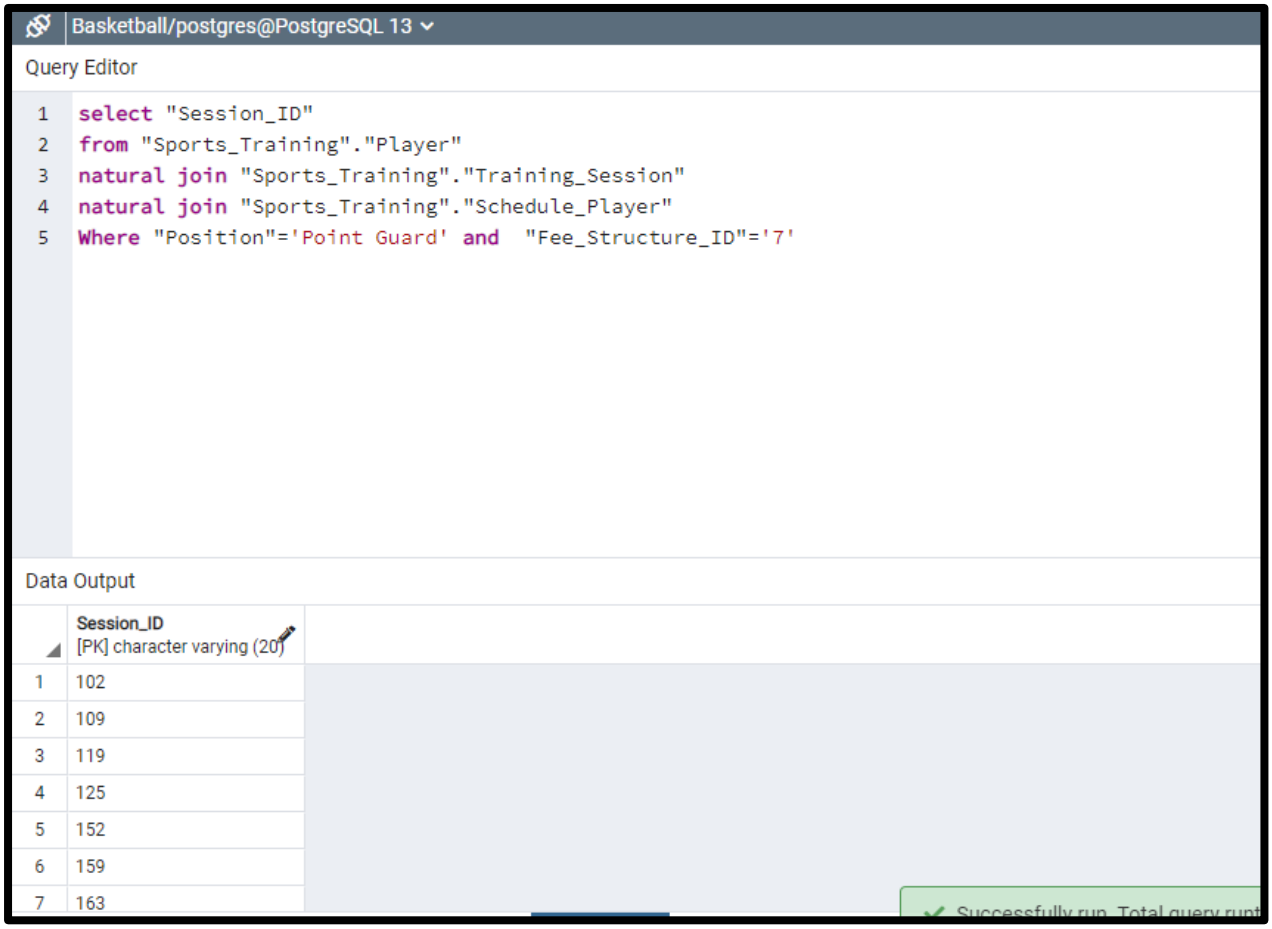
```
1. select "Name" from "Sports_Training"."Player" natural join "Sports_Training"."Grade_Card"
2. Where "Grade_Card_ID"='1' and "Shooting" > (select avg("Shooting") from
   "Sports_Training"."Grade_Card")
3. order by "Height"
```



Number of Records = 53

38. Find Session_ID of sessions attended by Players whose Position is Point Guard and Fee_Structure_ID is 7.

```
1. select "Session_ID"
2. from "Sports_Training"."Player"
3. natural join "Sports_Training"."Training_Session"
4. natural join "Sports_Training"."Schedule_Player"
5. Where "Position"='Point Guard' and  "Fee_Structure_ID"='7'
```



Number of Records = 15

39. Find the name of students who had session for Defence in Court number 9.

```
1. select "Name"
2. from "Sports_Training"."Player"
3. natural join "Sports_Training"."Training_Session"
4. natural join "Sports_Training"."Schedule_Player"
5. Where "Court_Number"='9' and  "Session_Type"='Defence'
```

Basketball/postgres@PostgreSQL 13

Query Editor

```
1  select "Name"
2  from "Sports_Training"."Player"
3  natural join "Sports_Training"."Training_Session"
4  natural join "Sports_Training"."Schedule_Player"
5  Where "Court_Number"='9' and  "Session_Type"='Defence'
```

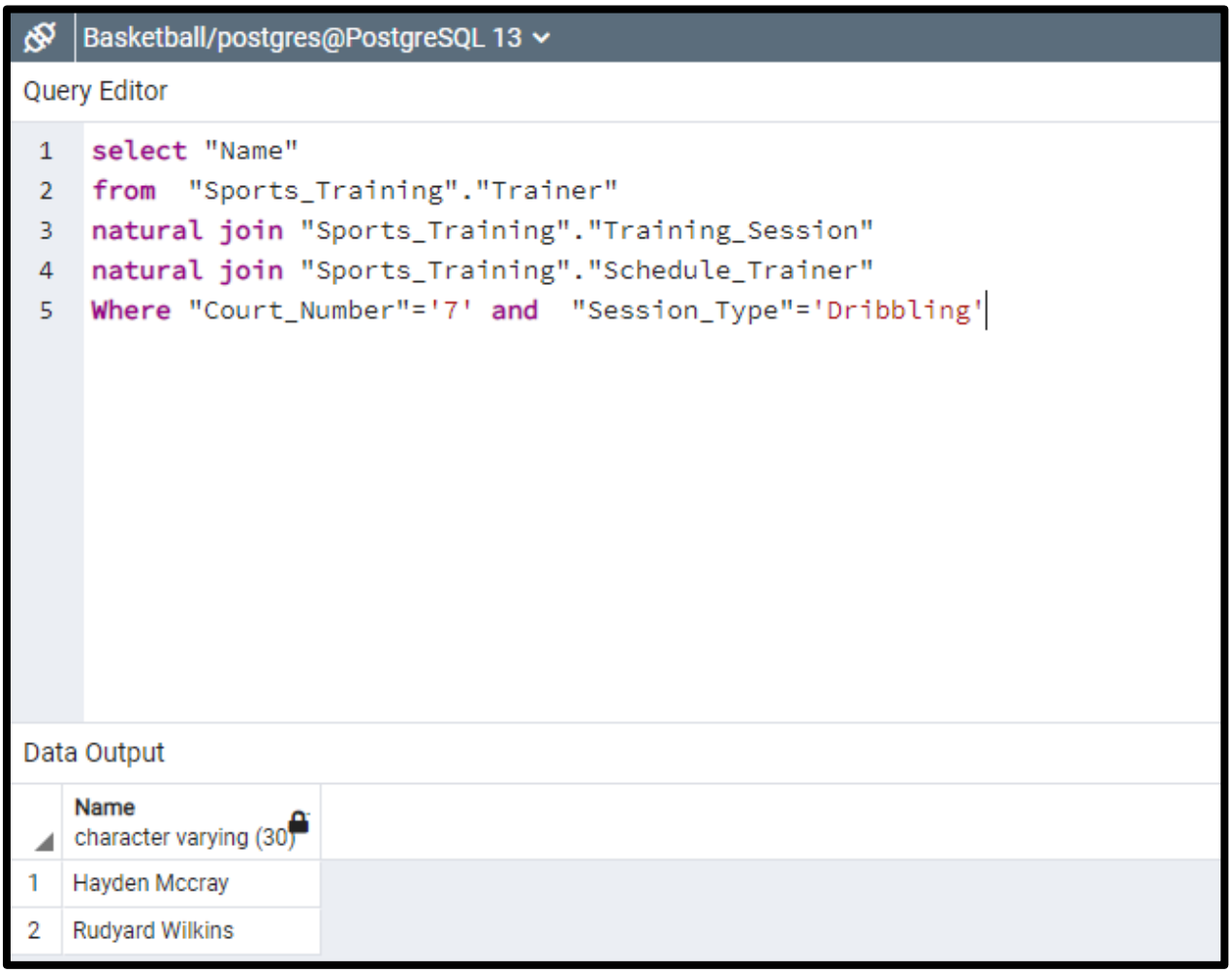
Data Output

	Name character varying (20)
1	Tatiana Bradford
2	Conan Shields
3	Oleg Mccarthy
4	Oleg Mccarthy
5	Wanda Riddle
6	Leroy Kent

Number of Records = 22

40. Find name of Trainer who had session for Dribbling in Court number 7.

```
1. select "Name"
2. from  "Sports_Training"."Trainer"
3. natural join "Sports_Training"."Training_Session"
4. natural join "Sports_Training"."Schedule_Trainer"
5. Where "Court_Number"='7' and  "Session_Type"='Dribbling'
```



Trigger Function:
Trigger Function To calculate Age from the Birth_Date.

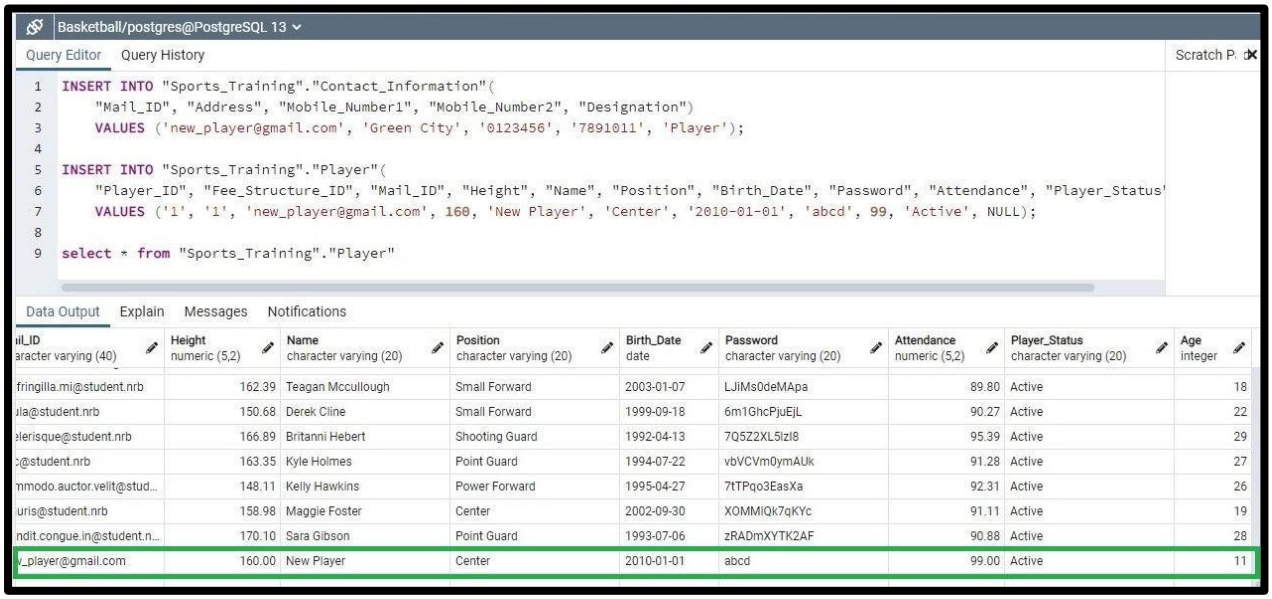
Code:

```
CREATE OR REPLACE FUNCTION "Sports_Training".age_1()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    VOLATILE
    COST 100
AS $BODY$
Declare
Begin
NEW."Age" := (Current_Date - NEW."Birth_Date")/365;
return new;
END
$BODY$;


CREATE TRIGGER age_1
    BEFORE INSERT OR UPDATE
    ON "Sports_Training"."Player"
    FOR EACH ROW
    EXECUTE FUNCTION "Sports_Training".age_1();
```

Demo Insert:

```
INSERT INTO "Sports_Training"."Contact_Information"(
    "Mail_ID", "Address", "Mobile_Number1", "Mobile_Number2", "Designation")
    VALUES ('new_player@gmail.com', 'Green City', '0123456', '7891011', 'Player');

INSERT INTO "Sports_Training"."Player"(
    "Player_ID", "Fee_Structure_ID", "Mail_ID", "Height", "Name", "Position", "Birth_Date",
"Password", "Attendance", "Player_Status", "Age")
    VALUES ('1', '1', 'new_player@gmail.com', 160, 'New Player', 'Center', '2010-01-01', 'abcd', 99,
'Active', NULL);

select * from "Sports_Training"."Player"
```

Screenshot:



Basketball/postgres@PostgreSQL 13

Query Editor | Query History | Scratch P...

```
1  INSERT INTO "Sports_Training"."Contact_Information"(
2      "Mail_ID", "Address", "Mobile_Number1", "Mobile_Number2", "Designation")
3      VALUES ('new_player@gmail.com', 'Green City', '0123456', '7891011', 'Player');
4
5  INSERT INTO "Sports_Training"."Player"(
6      "Player_ID", "Fee_Structure_ID", "Mail_ID", "Height", "Name", "Position", "Birth_Date", "Password", "Attendance", "Player_Status"
7      VALUES ('1', '1', 'new_player@gmail.com', 160, 'New Player', 'Center', '2010-01-01', 'abcd', 99, 'Active', NULL);
8
9  select * from "Sports_Training"."Player"
```

Data Output | Explain | Messages | Notifications

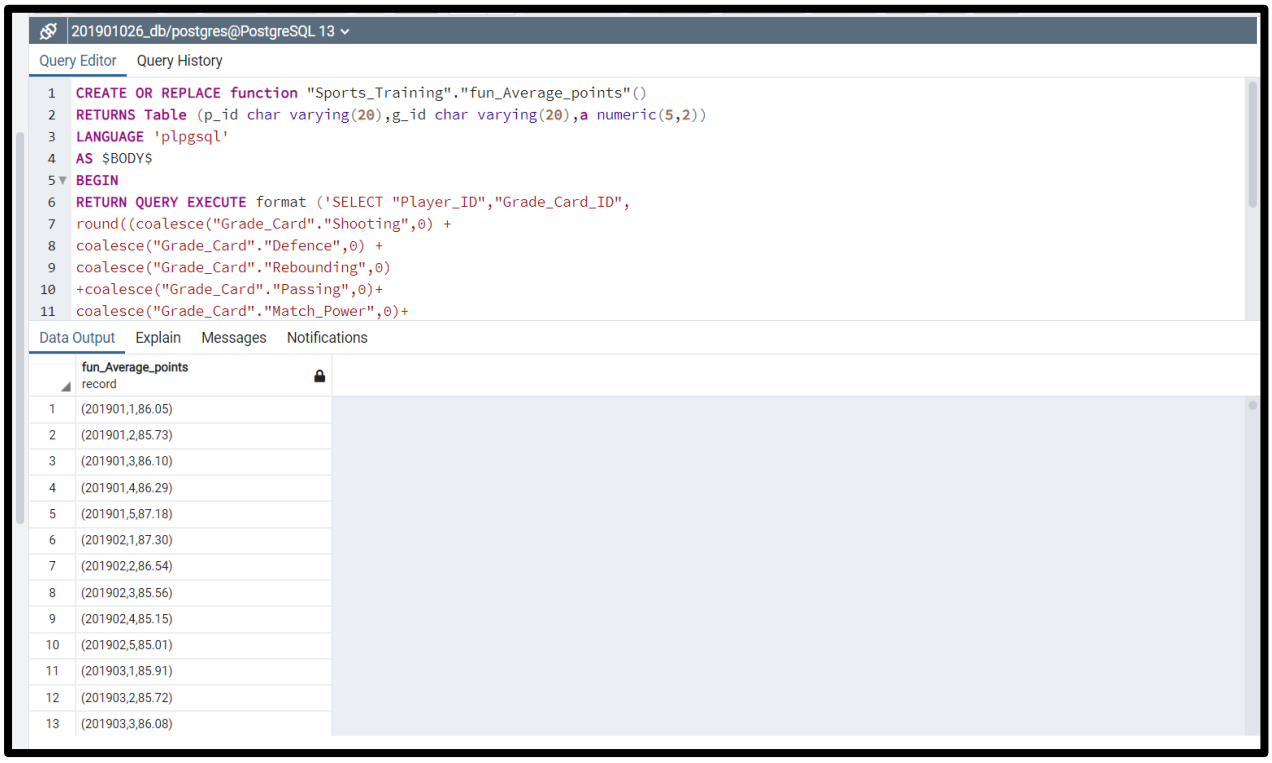
...il_ID ...aracter varying (40)	Height numeric (5,2)	Name character varying (20)	Position character varying (20)	Birth_Date date	Password character varying (20)	Attendance numeric (5,2)	Player_Status character varying (20)	Age integer
...fringilla.mi@student.nrb	162.39	Teagan Mccullough	Small Forward	2003-01-07	LJiMs0deMApa	89.80	Active	18
...ula@student.nrb	150.68	Derek Cline	Small Forward	1999-09-18	6m1GhcPjuEjL	90.27	Active	22
...elerisque@student.nrb	166.89	Britanni Hebert	Shooting Guard	1992-04-13	7Q5Z2XL5IzI8	95.39	Active	29
...@student.nrb	163.35	Kyle Holmes	Point Guard	1994-07-22	vbVCVm0ymAUk	91.28	Active	27
...mmodo.auctor.velit@stud...	148.11	Kelly Hawkins	Power Forward	1995-04-27	7tTPqo3EasXa	92.31	Active	26
...uris@student.nrb	158.98	Maggie Foster	Center	2002-09-30	XOMMlQk7qKYc	91.11	Active	19
...ndit.congue.in@student.n...	170.10	Sara Gibson	Point Guard	1993-07-06	zRADmXYTK2AF	90.88	Active	28
...w_player@gmail.com	160.00	New Player	Center	2010-01-01	abcd	99.00	Active	11

Function:

Function To calculate Average Points for all the Players with respect to their Grade Card IDs:

Query:

```
1.  CREATE OR REPLACE function "Sports_Training"."fun_Average_points"()
2.  RETURNS Table (p_id char varying(20),g_id char varying(20),a numeric(5,2))
3.  LANGUAGE 'plpgsql'
4.  AS $BODY$
5.  BEGIN
6.  RETURN QUERY EXECUTE format ('SELECT "Player_ID","Grade_Card_ID",
7.  round((coalesce("Grade_Card"."Shooting",0) +
8.  coalesce("Grade_Card"."Defence",0) +
9.  coalesce("Grade_Card"."Rebounding",0)
10. +coalesce("Grade_Card"."Passing",0)+
11. coalesce("Grade_Card"."Match_Power",0)+
12. coalesce("Grade_Card"."Fitness",0)+
13. coalesce("Grade_Card"."Dribbling",0))/7,2)
14. FROM "Sports_Training"."Grade_Card"
15. ');
16. END;
17. $BODY$;
18.
19.
20. Select "Sports_Training"."fun_Average_points"()
```



Number Of Tuples = 480

Section 7

7) Project Code with Output Screenshots

Codes:

We have made GUI using python library, PyWebIO. We have given 16 functions in our web application.

List of functions is as follows:

Insert

- Insert Contact Information
- Insert Player
- Insert Trainer

Delete

- Delete Contact Information
- Delete Player Information
- Delete Trainer Information

Show Table

- Show Contact Information
- Show Player
- Show Trainer

Search

- Search Player
- Search Contact Information

Query

- Average age of Player
- List all the player names of team who has scored highest points among other teams
- Find all player's names who played the highest matches
- List Date and Court Number for the matches won by team "Panthers"
- List all the player's names who played on 22 nd January, 2001 at court No. 12

File Details:

- main.py contains code for home page.
- back.py contains code for database connection and queries.
- front.py contains code for layout and input-output for each function.
- Codes and needed files are also uploaded at this drive link: https://drive.google.com/folderview?id=1o2G4RPbJiYBz72GUN_hjod783rIj_h6U
- Demo Video to describe the Front-End application: https://drive.google.com/file/d/1gei-MFe1O_kMQivzKwZhTr10kFL22O2C/view?usp=sharing

main.py:

```
def home_page():
    import front
    import pywebio.output as pwo
    from pywebio.input import input
    pwo.clear()
    pwo.put_html('''<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-image: url('https://images.unsplash.com/photo-1618005182384-a83a8bd57fbe?ixlib=rb-1.2.1&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=1528&q=80');
}
</style>
</head>
<body>
</body>
</html>
''' )
    pwo.set_scope('List')
    pwo.put_markdown(""" # Navrang Basketball Academy""")
    pwo.put_markdown(""" ## Insert""")
    pwo.put_buttons(['Insert Contact Information'], onclick = [front.in_info_f])
    pwo.put_buttons(['Insert Player'], onclick = [front.in_player_f])
    pwo.put_buttons(['Insert Trainer'], onclick = [front.in_trainer_f])
    pwo.put_markdown(""" ## Delete""")
    pwo.put_buttons(['Delete Contact Information'], onclick = [front.del_contact_info_f])
    pwo.put_buttons(['Delete Player Information'], onclick = [front.del_player_info_f])
    pwo.put_buttons(['Delete Trainer Information'], onclick = [front.del_trainer_info_f])
    pwo.put_markdown(""" ## Show Table""")
    pwo.put_buttons(['Show Contact Information'], onclick = [front.show_info_f])
    pwo.put_buttons(['Show Player'], onclick = [front.show_player_f])
    pwo.put_buttons(['Show Trainer'], onclick = [front.show_trainer_f])
    pwo.put_markdown(""" ## Search""")
    pwo.put_buttons(['Search Player '], onclick = [front.search_player_f])
    pwo.put_buttons(['Search Contact Information'], onclick = [front.search_contact_f])
    pwo.put_markdown(""" ## Query""")
    pwo.put_buttons(['Average age of Player'], onclick = [front.query5_f])
    pwo.put_buttons(['List all the player names of team who has scored highest points among other teams'], onclick = [front.query1_f])
    pwo.put_buttons(['Find all player's names who played the highest matches'], onclick = [front.query2_f])
    pwo.put_buttons(['List Date and Court Number for the matches won by team "Panthers"'], onclick = [front.query3_f])
    pwo.put_buttons(['List all the player's names who played on 22 nd January, 2001 at court No. 12'], onclick = [front.query4_f])
    while(True):
        pass

```

```
45. if __name__ == '__main__':
46.     home_page()
```

back.py:

```
1.  import psycopg2
2.
3.  #establishing the connection
4.  conn = psycopg2.connect(
5.    database="Basketball", user='postgres', password='admin', host='127.0.0.1', port= '5432'
6.  )
7.  #Creating a cursor object using the cursor() method
8.  cursor = conn.cursor()
9.
10. #Setting auto commit false
11. conn.autocommit = True
12. def in_info(mail, address, m1, m2 ,des):
13.     conn = psycopg2.connect(database="Basketball", user='postgres', password='admin',
    host='127.0.0.1', port= '5432')
14.     cursor = conn.cursor()
15.     conn.autocommit = True
16.
17.     temp='''INSERT INTO "Sports_Training"."Contact_Information"(
18.     "Mail_ID", "Address", "Mobile_Number1", "Mobile_Number2", "Designation")
19.     VALUES (%s, %s, %s, %s, %s);'''
20.     data=[(mail,address,m1,m2,des)]
21.     try:
22.         cursor.executemany(temp, data)
23.         return 0
24.     except:
25.         return -1
26.     #Commit your changes in the database
27.
28. def in_player(pid,fid,mid,h,name,pos,dob, pw, att, pstatus,age):
29.     conn = psycopg2.connect(database="Basketball", user='postgres', password='admin',
    host='127.0.0.1', port= '5432')
30.     cursor = conn.cursor()
31.     conn.autocommit = True
32.     temp= '''INSERT INTO "Sports_Training"."Player"(
33.     "Player_ID", "Fee_Structure_ID", "Mail_ID", "Height", "Name", "Position", "Birth_Date",
    "Password", "Attendance", "Player_Status", "Age")
34.     VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s);'''
35.     data=[(pid,fid,mid,h,name,pos,dob, pw, att, pstatus,age)]
36.     try:
37.         cursor.executemany(temp, data)
38.         return 0
39.     except:
40.         return -1
```

```
41.
42.  def in_trainer(tid,mid,ssid,name,pw,att):
43.      conn = psycopg2.connect(database="Basketball", user='postgres', password='admin',
     host='127.0.0.1', port= '5432')
44.      cursor = conn.cursor()
45.      conn.autocommit = True
46.      temp='"INSERT INTO "Sports_Training"."Trainer"(
47.      "Trainer_ID", "Mail_ID", "Salary_Structure_ID", "Name", "Password", "Attendance")
48.      VALUES (%s, %s, %s, %s, %s, %s);'"
49.      data=[(tid,mid,ssid,name,pw,att)]
50.      try:
51.          cursor.executemany(temp, data)
52.          return 0
53.      except:
54.          return -1
55.
56.  def del_contact_info(mid):
57.      conn = psycopg2.connect(database="Basketball", user='postgres', password='admin',
     host='127.0.0.1', port= '5432')
58.      cursor = conn.cursor()
59.      conn.autocommit = True
60.      s = "DELETE FROM \"Sports_Training\".\"Contact_Information\" WHERE
     \"Mail_ID\"='{}';".format(mid)
61.      print(s)
62.      #Deleting records
63.      try:
64.          cursor.execute(s)
65.          print(cursor.rowcount)
66.          if(cursor.rowcount):
67.              return 0
68.          return -1
69.      except:
70.          return -1
71.
72.  def del_player_info(pid):
73.      conn = psycopg2.connect(database="Basketball", user='postgres', password='admin',
     host='127.0.0.1', port= '5432')
74.      cursor = conn.cursor()
75.      conn.autocommit = True
76.      s="DELETE FROM \"Sports_Training\".\"Player\" WHERE \"Player_ID\"='{}';".format(pid)
77.      #Deleting records
78.      try:
79.          cursor.execute(s)
80.          print(cursor.rowcount)
81.          if(cursor.rowcount):
82.              return 0
83.          return -1
84.      except:
85.          return -1
86.
87.  def del_trainer_info(tid):
```

```
88.     conn = psycopg2.connect(database="Basketball", user='postgres', password='admin',
    host='127.0.0.1', port= '5432')
89.     cursor = conn.cursor()
90.     conn.autocommit = True
91.     s = "DELETE FROM \"Sports_Training\".\"Trainer\" WHERE \"Trainer_ID\"='{}';".format(tid)
92.     print(s)
93.     #Deleting records
94.     try:
95.         cursor.execute(s)
96.         print(cursor.rowcount)
97.         if(cursor.rowcount):
98.             return 0
99.         return -1
100.    except:
101.        return -1
102.
103.def query1(): #query 31 List all the player names of team who has scored highest points among other
    teams.
104.    conn = psycopg2.connect(database="Basketball", user='postgres', password='admin',
    host='127.0.0.1', port= '5432')
105.    cursor = conn.cursor()
106.    conn.autocommit = True
107.    cursor.execute(''' Select "Name"
108.    From "Sports_Training"."Player"
109.    Where "Player_ID" in
110.    (Select "Player_ID"
111.    From "Sports_Training"."Association"
112.    where "Team_ID"=
113.    (Select "Team_ID"
114.    From "Sports_Training"."Team"
115.    Where "Team_Points"=(Select max("Team_Points")
116.    From "Sports_Training"."Team"))) and "Player_Status"='Active'  ''')
117.    result = cursor.fetchall();
118.    result.insert(0, ('Name',));
119.    return result
120.
121.
122.def query2(): #query 27
123.    conn = psycopg2.connect(database="Basketball", user='postgres', password='admin',
    host='127.0.0.1', port= '5432')
124.    cursor = conn.cursor()
125.    conn.autocommit = True
126.    cursor.execute(''' select distinct "Name"
127.    from "Sports_Training"."Player"
128.    natural join "Sports_Training"."Association"
129.    where "Matches_Played" =(select max("Matches_Played")
130.    from "Sports_Training"."Association")''')
131.    result = cursor.fetchall();
132.    result.insert(0, ('Name',));
133.    return result
134.
135.def query3(): #query 32
```

```
    conn = psycopg2.connect(database="Basketball", user='postgres', password='admin', host='127.0.0.1', port= '5432')
    cursor = conn.cursor()
    conn.autocommit = True
    cursor.execute(''' select "Date", "Court_Number"
    from ("Sports_Training"."Team" natural join "Sports_Training"."Team_Match" natural join "Sports_Training"."Matches" natural join "Sports_Training"."Training_Session") as rec
    where "Winning_Team_Name" = 'Panthers'
    group by "Court_Number", "Date"
    order by "Date" ''')
    result = cursor.fetchall();
    result.insert(0,('Date' , 'Court_Number'));
    return result

def query4(): #query 35
    conn = psycopg2.connect(database="Basketball", user='postgres', password='admin', host='127.0.0.1', port= '5432')
    cursor = conn.cursor()
    conn.autocommit = True
    cursor.execute(''' Select "Name"
    From "Sports_Training"."Player"
    Where "Player_ID" in(Select "Player_ID"
    From "Sports_Training"."Schedule_Player"
    Where "Session_ID"=(Select "Session_ID"
    From "Sports_Training"."Training_Session"
    Where "Date"='22-01-2021' and "Court_Number"=12))''')
    result = cursor.fetchall();
    result.insert(0, ('Name',));
    return result

def query5():
    conn = psycopg2.connect(database="Basketball", user='postgres', password='admin', host='127.0.0.1', port= '5432')
    cursor = conn.cursor()
    conn.autocommit = True
    cursor.execute(''' select vsround(avg("Age"),2) from "Sports_Training"."Player"''')
    result = cursor.fetchall();
    result.insert(0, ('Average Age',));
    return result

def show_info():
    conn = psycopg2.connect(database="Basketball", user='postgres', password='admin', host='127.0.0.1', port= '5432')
    cursor = conn.cursor()
    conn.autocommit = True
    cursor.execute('''SELECT * FROM "Sports_Training"."Contact_Information";''')
    result = cursor.fetchall();
    result.insert(0, ('Mail_ID', 'Address', 'Mobile_Number1', 'Mobile_Number2', 'Designation'));
    return (result)

def show_player():
```

```
    conn = psycopg2.connect(database="Basketball", user='postgres', password='admin', host='127.0.0.1', port= '5432')
    cursor = conn.cursor()
    conn.autocommit = True
    cursor.execute('"SELECT * FROM "Sports_Training"."Player";"')
    result = cursor.fetchall();
    result.insert(0, ('Player_ID', 'Fee_Structure_ID', 'Mail_ID', 'Height', 'Name', 'Position ', 'Birth_Date' , 'Password' , 'Attendance' , 'Player_Status' , 'Age'));
    return (result)

def show_trainer():
    conn = psycopg2.connect(database="Basketball", user='postgres', password='admin', host='127.0.0.1', port= '5432')
    cursor = conn.cursor()
    conn.autocommit = True
    cursor.execute('"SELECT * FROM "Sports_Training"."Trainer";"')
    result = cursor.fetchall();
    result.insert(0, ('Trainer_ID', 'Mail_ID', 'Salary_Structure_ID', 'Name', 'Password' , 'Attendance' ));
    return (result)

def search_player(p_id):
    conn = psycopg2.connect(database="Basketball", user='postgres', password='admin', host='127.0.0.1', port= '5432')
    cursor = conn.cursor()
    conn.autocommit = True
    s = 'Select * From "Sports_Training"."Player" Where "Player_ID"=\'{}\';'.format(p_id)
    print(s)
    try:
        cursor.execute(s)
        ans = cursor.fetchall()
        ans.insert(0, ('Player_ID', 'Fee_Structure_ID', 'Mail_ID', 'Height', 'Name', 'Position ', 'Birth_Date' , 'Password' , 'Attendance' , 'Player_Status' , 'Age'));
        return ans
    except:
        return -1

def search_contact(mail_id):
    conn = psycopg2.connect(database="Basketball", user='postgres', password='admin', host='127.0.0.1', port= '5432')
    cursor = conn.cursor()
    conn.autocommit = True
    s = 'Select * From "Sports_Training"."Contact_Information" Where "Mail_ID"=\'{}\';'.format(mail_id)
    print(s)
    try:
        cursor.execute(s)
        ans = cursor.fetchall()
        ans.insert(0, ('Mail_ID', 'Address', 'Mobile number 1', 'Mobile number 2', 'Designation'));
        return ans
    except:
        return -1

```

```
227.
228.
229.conn.commit()
230.
231.# Closing the connection
232.conn.close()
```

front.py:

```
1.  from back import *
2.  import pywebio.output as pwo
3.  from pywebio.input import *
4.  from main import *
5.
6.  def in_info_f():
7.      pwo.clear()
8.      arr = input_group("Insert Contact Information",[
9.      input('Mail_ID', name='name'),
10.     input('Address', name='addrs'),
11.     input('Mobile Number 1', name='m1'),
12.     input('Mobile Number 2', name='m2'),
13.     input('Designation', name='desig')])
14.     isSuccessful = in_info(str(arr['name']), str(arr['addrs']), str(arr['m1']), str(arr['m2']), str(arr['desig']))
15.     if(isSuccessful == -1):
16.         img=open("something-went-wrong_f.gif",'rb').read()
17.         pwo.put_image(img)
18.     else:
19.         img=open("inset_successful_f.gif",'rb').read()
20.         pwo.put_image(img)
21.     pwo.put_buttons(['Home Page'], onclick = [home_page])
22.
23. def in_player_f():
24.     pwo.clear()
25.     arr = input_group("Insert Player Information",[
26.     input('Player_ID', name='1'),
27.     input('Fee_Structure_ID', name='2', type=NUMBER),
28.     input('Mail_ID', name='3'),
29.     input('Height', name='4'),
30.     input('Name', name='5'),
31.     input('Position', name = '6'),
32.     input('Birth_Date', name = '7', type=DATE),
33.     input('Password', name='8', type = PASSWORD),
34.     input('Attendance', name='9'),
35.     input('Player_Status', name='10')])
36.     arr['11'] = 0
37.     isSuccessful = in_player(arr['1'], arr['2'], arr['3'], arr['4'], arr['5'], arr['6'], arr['7'], arr['8'], arr['9'],
    arr['10'], arr['11'])
38.     if(isSuccessful == -1):
```

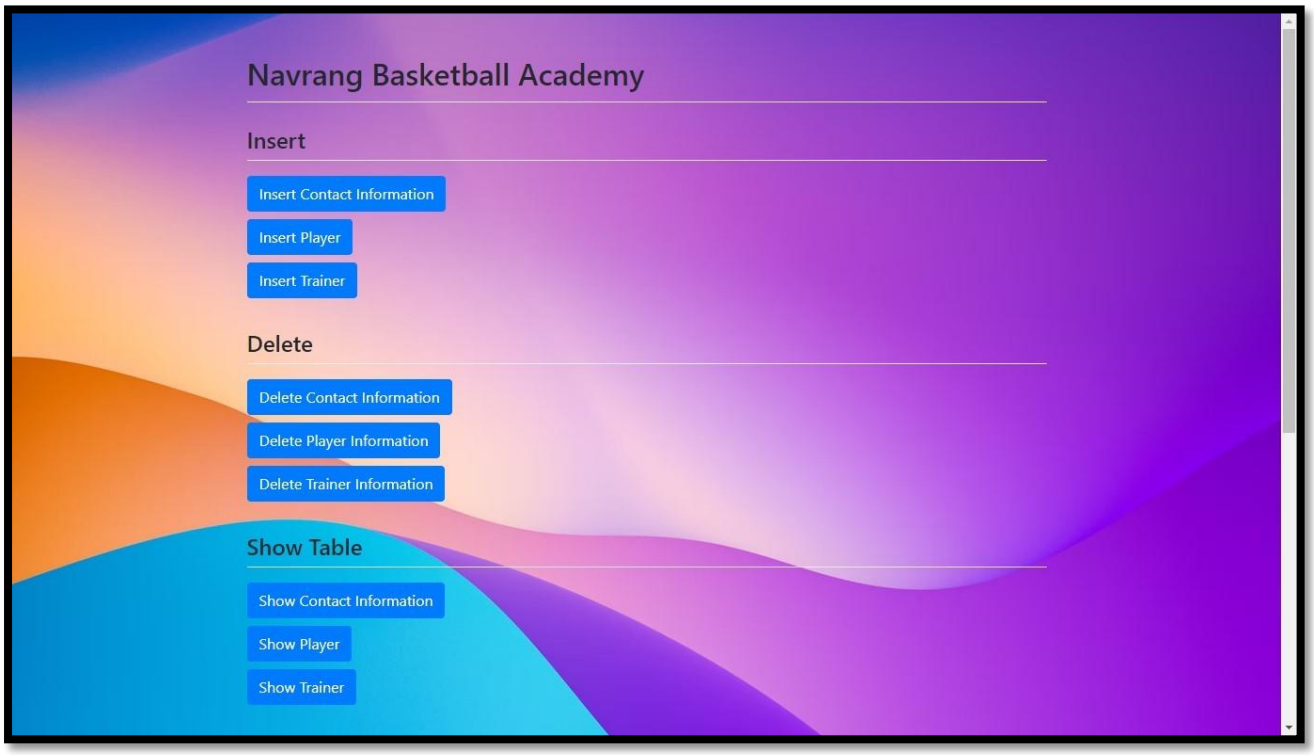
```
        img=open("something-went-wrong_f.gif",'rb').read()
        pwo.put_image(img)
    else:
        img=open("inset_successful_f.gif",'rb').read()
        pwo.put_image(img)
    pwo.put_buttons(['Home Page'], onclick = [home_page])


def in_trainer_f():
    pwo.clear()
    arr = input_group("Insert Trainer Information",[
    input('Trainer_ID', name='1'),
    input('Mail_ID', name='2'),
    input('Salary_Structure_ID', name='3', type=NUMBER),
    input('Name', name='4'),
    input('Password', name='5', type = PASSWORD),
    input('Attendance', name='6', type = FLOAT)])
    isSuccessful = in_trainer(arr['1'], arr['2'], arr['3'], arr['4'], arr['5'], arr['6'])
    if(isSuccessful == -1):
        img=open("something-went-wrong_f.gif",'rb').read()
        pwo.put_image(img)
    else:
        img=open("inset_successful_f.gif",'rb').read()
        pwo.put_image(img)
    pwo.put_buttons(['Home Page'], onclick = [home_page])

def del_contact_info_f():
    pwo.clear()
    s = input("Insert Primary key: ")
    del1 = del_contact_info(s)
    if(del1 == -1):
        img=open("something-went-wrong_f.gif",'rb').read()
        pwo.put_image(img)
    else:
        img=open("delete_successful_f.gif",'rb').read()
        pwo.put_image(img)
    pwo.put_buttons(['Home Page'], onclick = [home_page])

def del_player_info_f():
    pwo.clear()
    s = input("Insert Primary key: ")
    del1 = del_player_info(s)
    if(del1 == -1):
        img=open("something-went-wrong_f.gif",'rb').read()
        pwo.put_image(img)
    else:
        img=open("delete_successful_f.gif",'rb').read()
        pwo.put_image(img)
    pwo.put_buttons(['Home Page'], onclick = [home_page])

def del_trainer_info_f():
    pwo.clear()
```

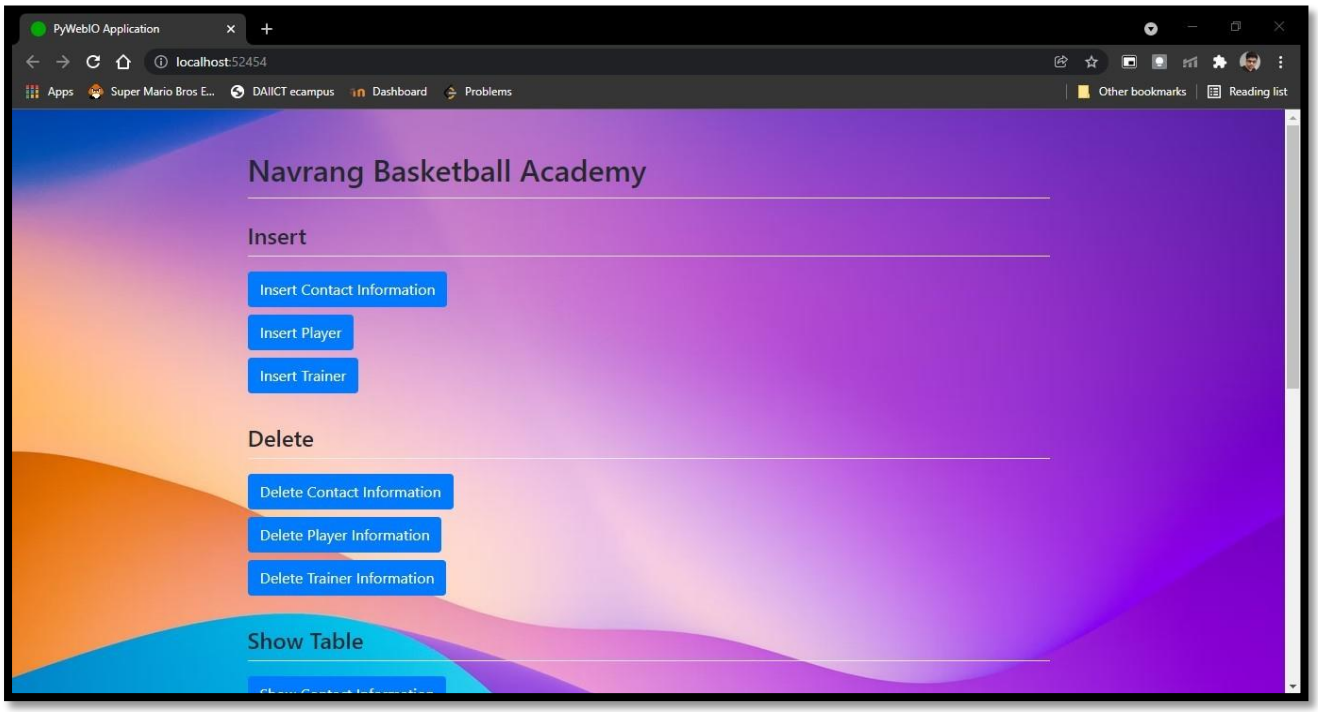
```
    s = input("Insert Primary key: ")
    del1 = del_trainer_info(s)
    if(del1 == -1):
        img=open("something-went-wrong_f.gif",'rb').read()
        pwo.put_image(img)
    else:
        img=open("delete_successful_f.gif",'rb').read()
        pwo.put_image(img)
    pwo.put_buttons(['Home Page'], onclick = [home_page])

def show_info_f():
    pwo.clear()
    txt = show_info()
    pwo.put_markdown(""" # Contact Information""")
    pwo.put_table(txt)
    pwo.put_buttons(['Home Page'], onclick = [home_page])

def show_player_f():
    pwo.clear()
    txt = show_player()
    pwo.put_markdown(""" # Player""")
    pwo.put_table(txt)
    pwo.put_buttons(['Home Page'], onclick = [home_page])

def show_trainer_f():
    pwo.clear()
    txt = show_trainer()
    pwo.put_markdown(""" # Trainer""")
    pwo.put_table(txt)
    pwo.put_buttons(['Home Page'], onclick = [home_page])

def search_player_f():
    pwo.clear()
    p_id=int(input("Enter Player_ID"))
    ans=search_player(p_id);
    if(ans!=-1):
        pwo.put_markdown(""" # Player Information""")

        pwo.put_table(ans)
    elif ans==-1:
        img=open("something-went-wrong_f.gif",'rb').read()
        pwo.put_image(img)
    pwo.put_buttons(['Home Page'], onclick = [home_page])

def search_contact_f():
    pwo.clear()
    mail_id=input("Enter Mail_ID")
    ans=search_contact(mail_id);
    if(ans!=-1):
        pwo.put_markdown(""" # Contact Information""")
        pwo.put_table(ans)
    elif ans==-1:
```

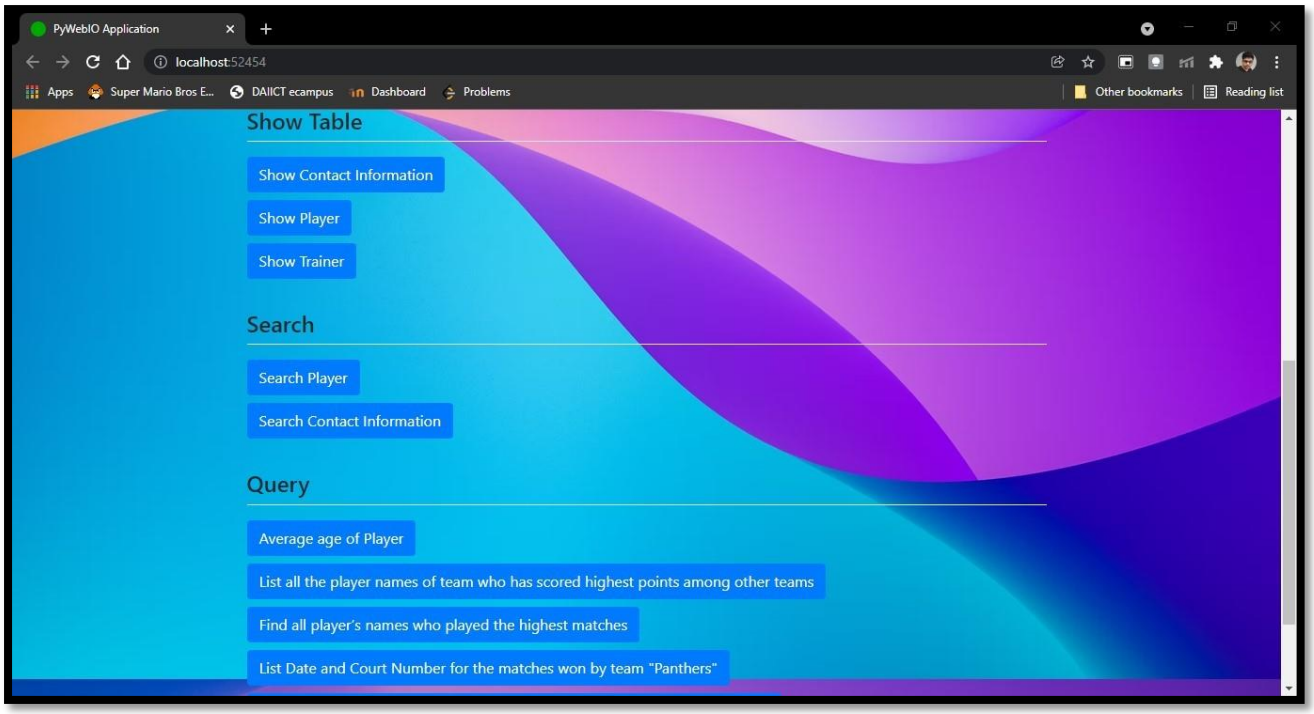
```
143.        img=open("something-went-wrong_f.gif",'rb').read()
144.        pwo.put_image(img)
145.    pwo.put_buttons(['Home Page'], onclick = [home_page])
146.
147.def query1_f():
148.    pwo.clear()
149.    txt = query1()
150.    pwo.put_markdown(""" # List all the player names of team who has scored highest points among
        other teams.""")
151.    pwo.put_table(txt)
152.    pwo.put_buttons(['Home Page'], onclick = [home_page])
153.
154.def query2_f():
155.    pwo.clear()
156.    txt = query2()
157.    pwo.put_markdown(""" # Find all player's names who played the highest matches.""")
158.    pwo.put_table(txt)
159.    pwo.put_buttons(['Home Page'], onclick = [home_page])
160.
161.def query3_f():
162.    pwo.clear()
163.    txt = query3()
164.    pwo.put_markdown(""" # List Date and Court Number for the matches won by team "Panthers".""")
165.    pwo.put_table(txt)
166.    pwo.put_buttons(['Home Page'], onclick = [home_page])
167.
168.def query4_f():
169.    pwo.clear()
170.    txt = query4()
171.    pwo.put_markdown(""" # List all the player's names who played on 22 nd January, 2001 at court No.
        12.""")
172.    pwo.put_table(txt)
173.    pwo.put_buttons(['Home Page'], onclick = [home_page])
174.
175.def query5_f():
176.    pwo.clear()
177.    txt = query5()
178.    pwo.put_markdown(""" # Find the Average Age of all Player. """)
179.    pwo.put_table(txt)
180.    pwo.put_buttons(['Home Page'], onclick = [home_page])
181.
```

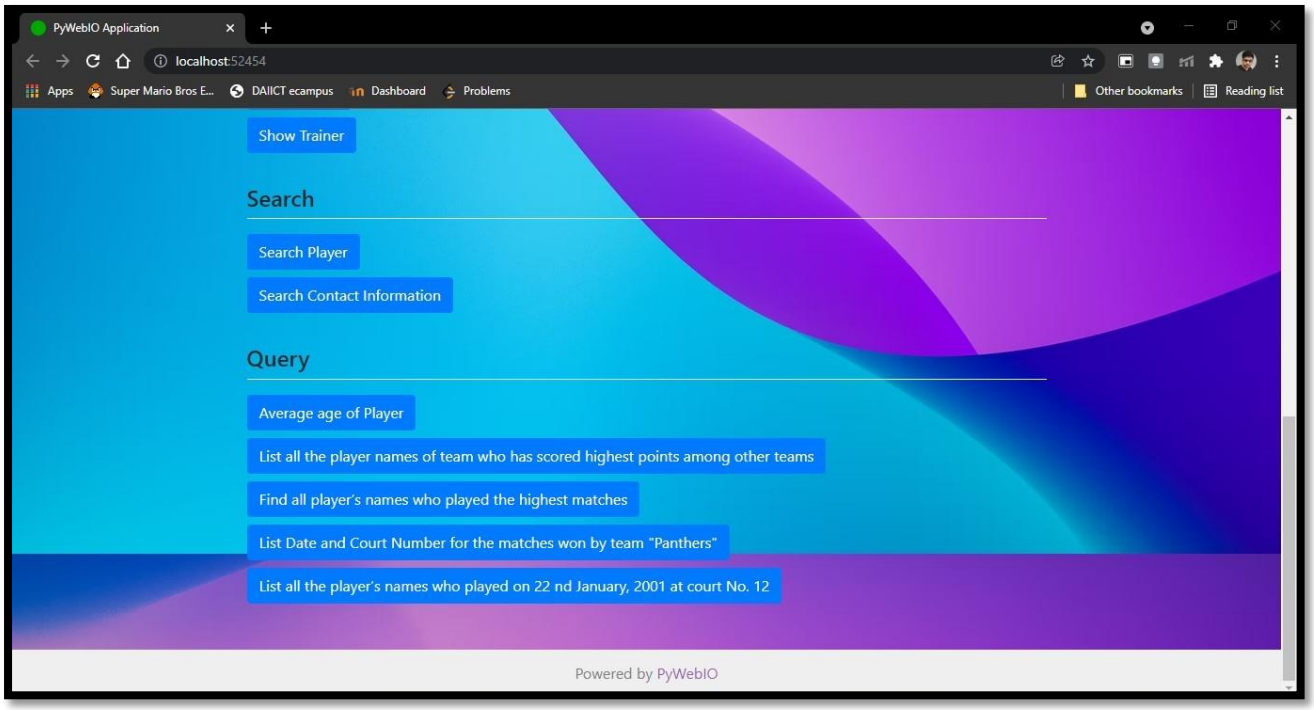
Screenshots:

Home Page:



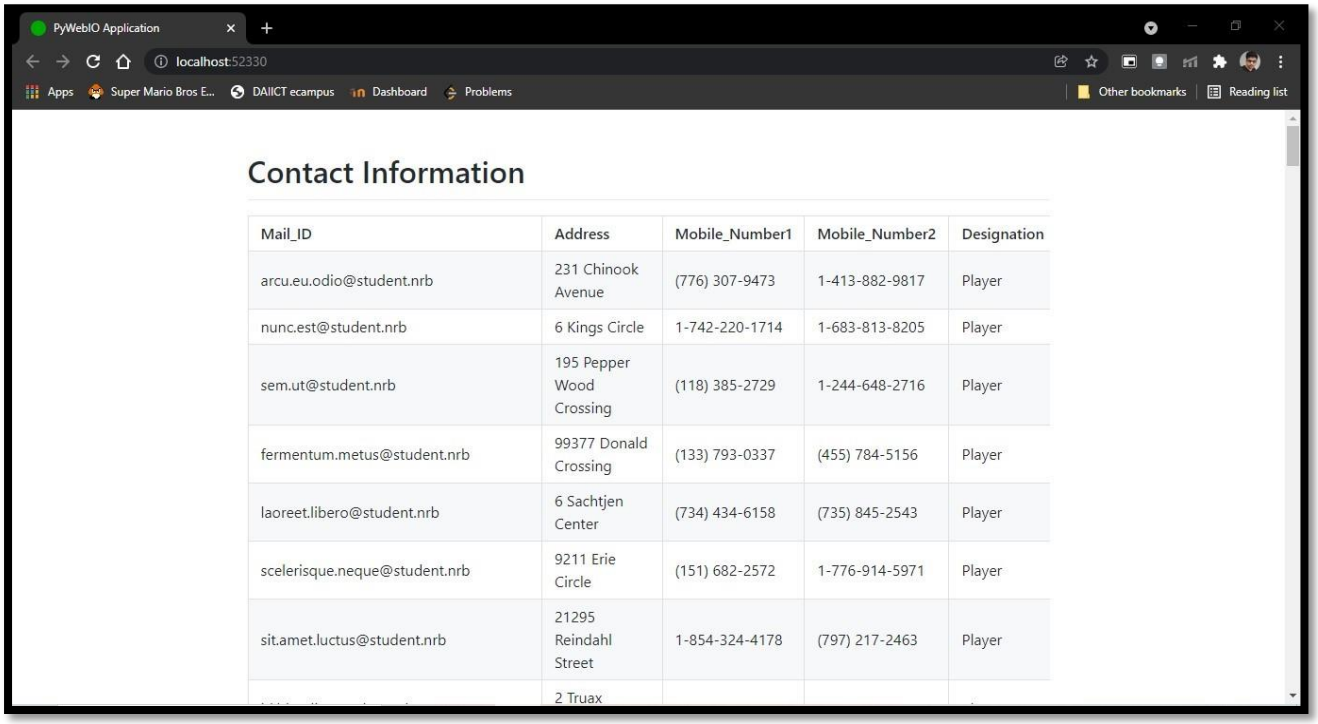


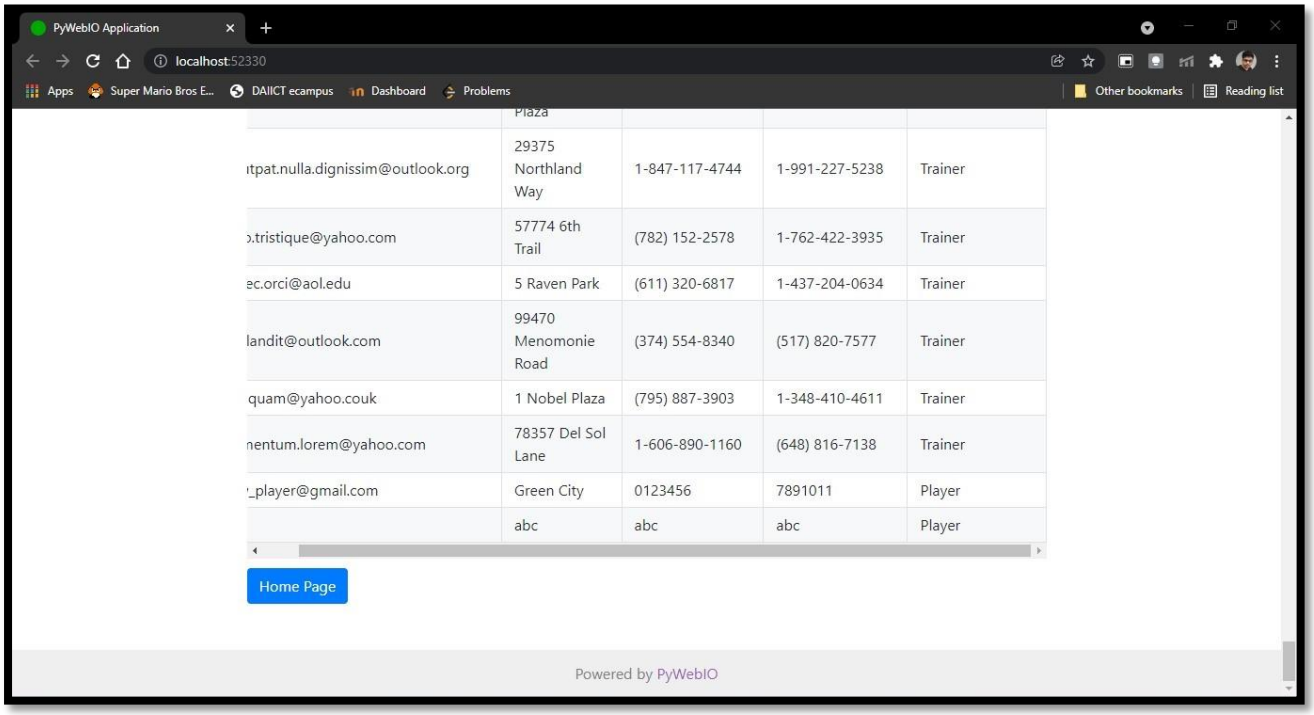

PyWebIO Application
localhost:52454
Apps
Super Mario Bros E...
DAIICT ecampus
Dashboard
Problems
Other bookmarks
Reading list
Show Table
Show Contact Information
Show Player
Show Trainer
Search
Search Player
Search Contact Information
Query
Average age of Player
List all the player names of team who has scored highest points among other teams
Find all player's names who played the highest matches
List Date and Court Number for the matches won by team "Panthers"


PyWebIO Application
localhost:52454
Apps
Super Mario Bros E...
DAIICT ecampus
Dashboard
Problems
Other bookmarks
Reading list
Show Trainer
Search
Search Player
Search Contact Information
Query
Average age of Player
List all the player names of team who has scored highest points among other teams
Find all player's names who played the highest matches
List Date and Court Number for the matches won by team "Panthers"
List all the player's names who played on 22 nd January, 2001 at court No. 12
Powered by PyWebIO

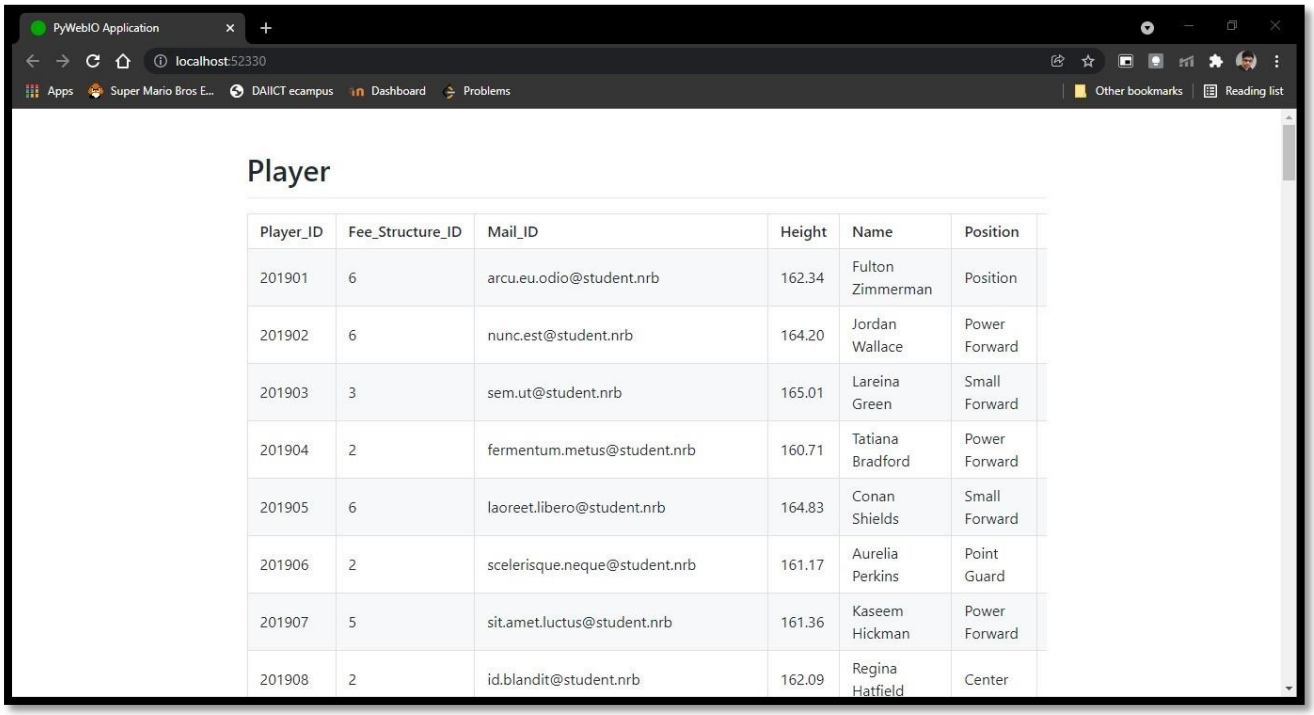
Show Table:

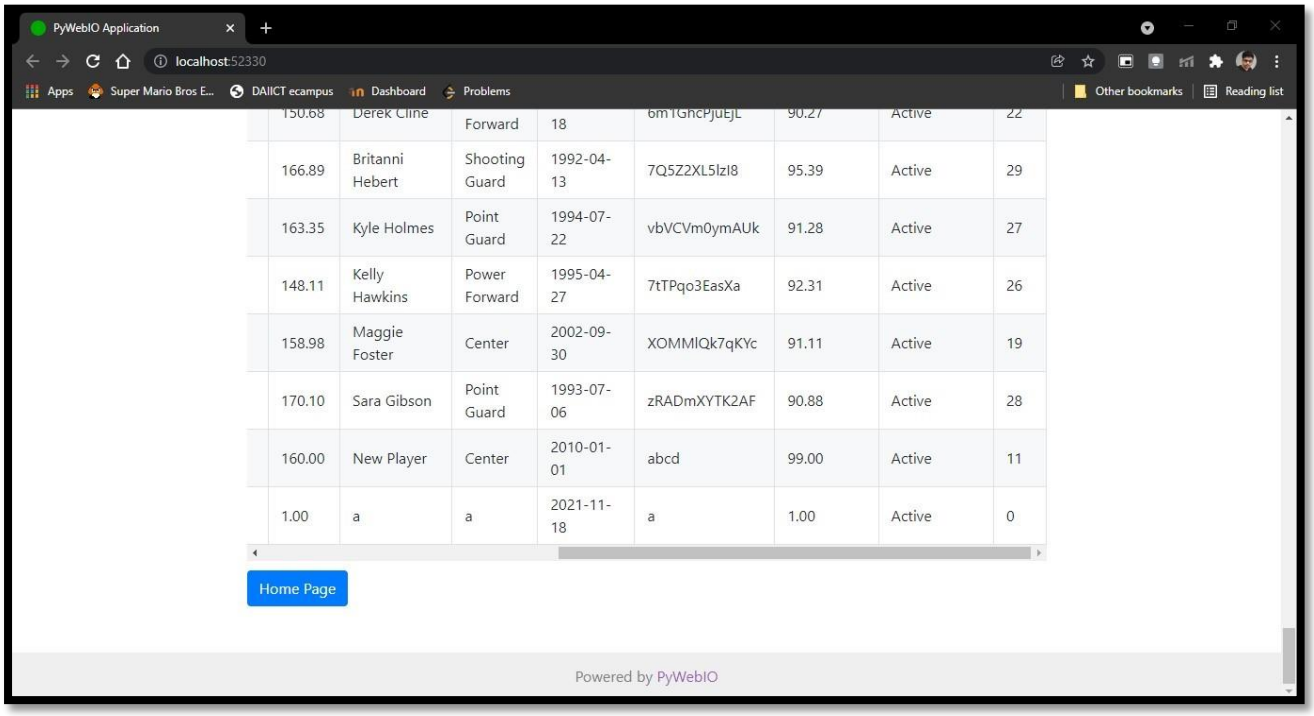
Contact Information:



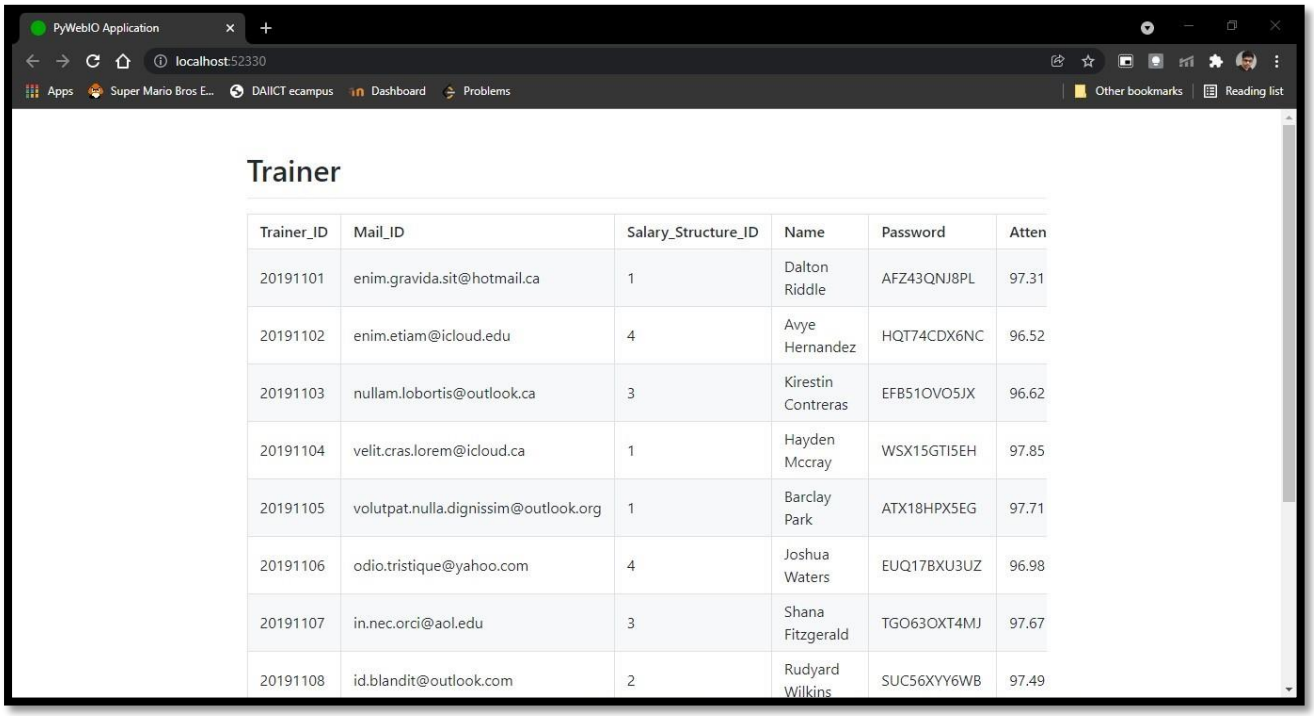


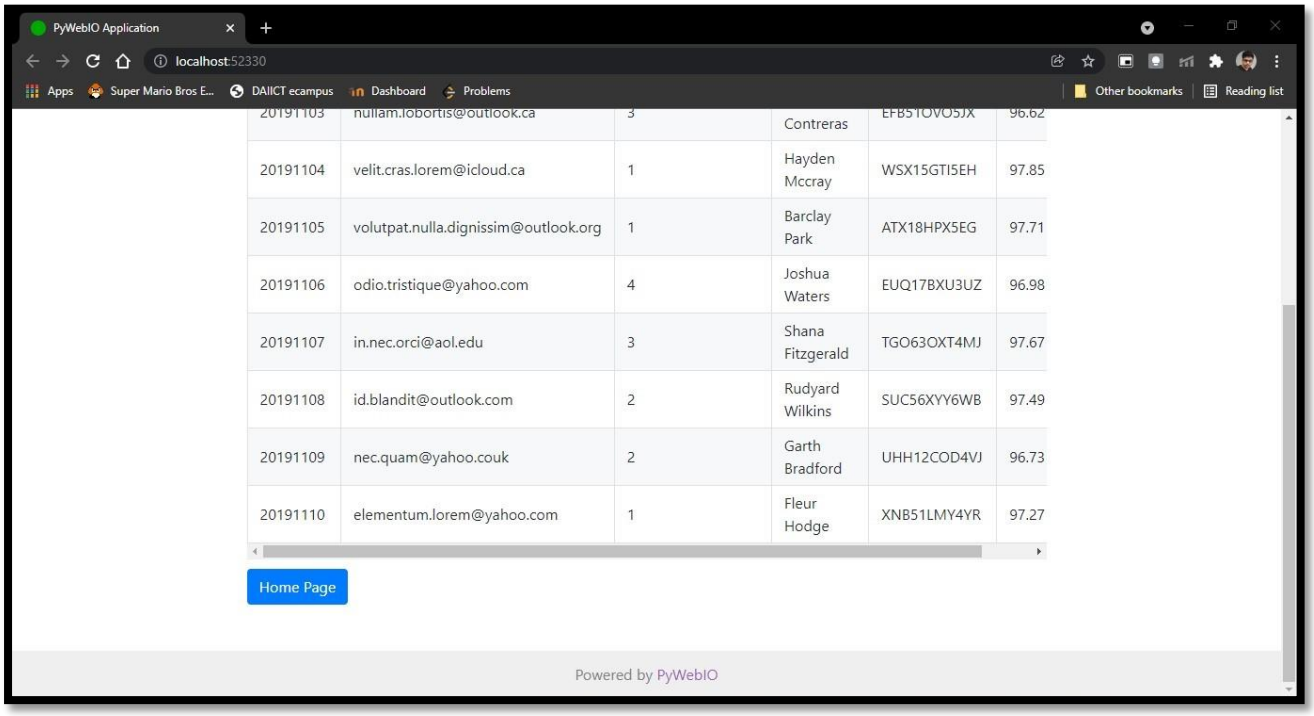
Player:


PyWebIO Application
localhost:52330
Apps
Super Mario Bros E...
DAIICT ecampus
Dashboard
Problems
Other bookmarks
Reading list
Player
Player_ID | Fee_Structure_ID | Mail_ID | Height | Name | Position
201901 | 6 | arcu.eu.odio@student.nrb | 162.34 | Fulton Zimmerman | Position
201902 | 6 | nunc.est@student.nrb | 164.20 | Jordan Wallace | Power Forward
201903 | 3 | sem.ut@student.nrb | 165.01 | Lareina Green | Small Forward
201904 | 2 | fermentum.metus@student.nrb | 160.71 | Tatiana Bradford | Power Forward
201905 | 6 | laoreet.libero@student.nrb | 164.83 | Conan Shields | Small Forward
201906 | 2 | scelerisque.neque@student.nrb | 161.17 | Aurelia Perkins | Point Guard
201907 | 5 | sit.amet.luctus@student.nrb | 161.36 | Kaseem Hickman | Power Forward
201908 | 2 | id.blandit@student.nrb | 162.09 | Regina Hatfield | Center


PyWebIO Application
localhost:52330
Apps
Super Mario Bros E...
DAIICT ecampus
Dashboard
Problems
Other bookmarks
Reading list
150.68 | Derek Cline | Forward | 18 | 6m1GhcPjuEjL | 90.27 | Active | 22
166.89 | Britanni Hebert | Shooting Guard | 1992-04-13 | 7Q5Z2XL5lzI8 | 95.39 | Active | 29
163.35 | Kyle Holmes | Point Guard | 1994-07-22 | vbVCVm0ymAUk | 91.28 | Active | 27
148.11 | Kelly Hawkins | Power Forward | 1995-04-27 | 7tTPqo3EasXa | 92.31 | Active | 26
158.98 | Maggie Foster | Center | 2002-09-30 | XOMMlQk7qKYc | 91.11 | Active | 19
170.10 | Sara Gibson | Point Guard | 1993-07-06 | zRADmXYTK2AF | 90.88 | Active | 28
160.00 | New Player | Center | 2010-01-01 | abcd | 99.00 | Active | 11
1.00 | a | a | 2021-11-18 | a | 1.00 | Active | 0
Home Page
Powered by PyWebIO

Trainer:

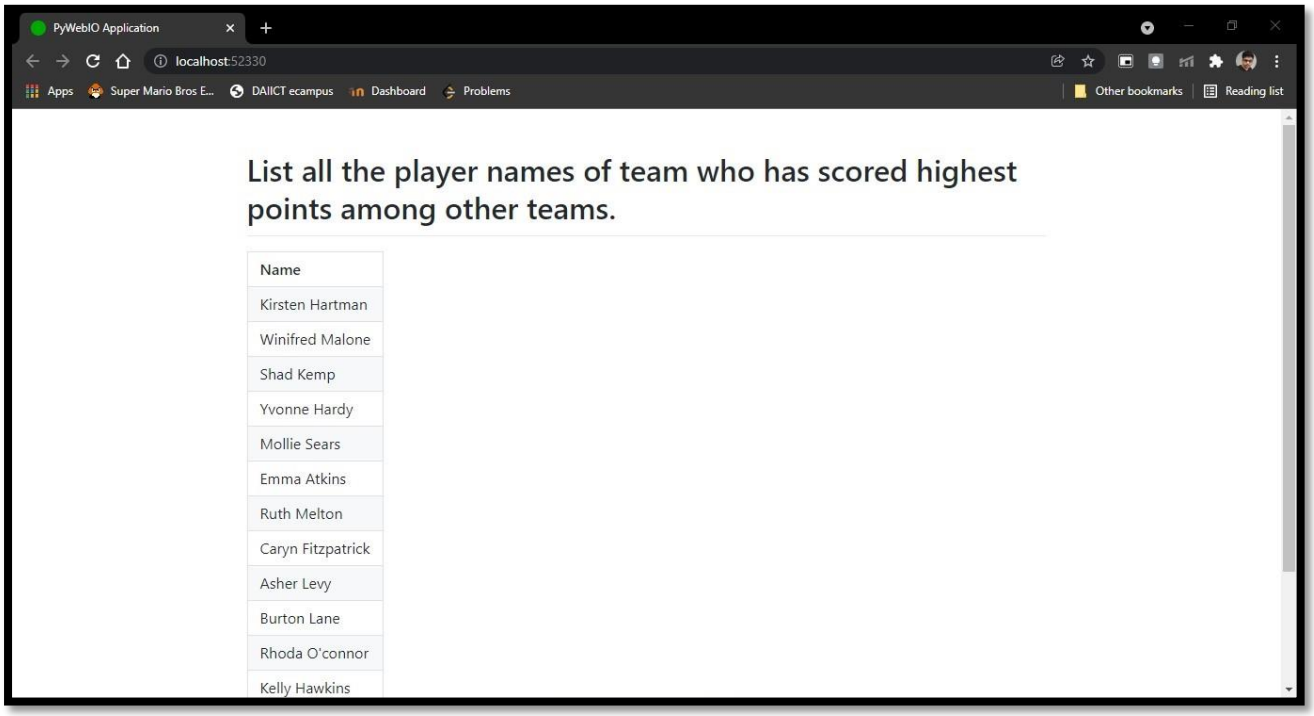

PyWebIO Application
localhost:52330
Apps
Super Mario Bros E...
DAIICT ecampus
Dashboard
Problems
Other bookmarks
Reading list
Trainer
Trainer_ID | Mail_ID | Salary_Structure_ID | Name | Password | Atten
20191101 | enim.gravida.sit@hotmail.ca | 1 | Dalton Riddle | AFZ43QNJ8PL | 97.31
20191102 | enim.etiam@icloud.edu | 4 | Avye Hernandez | HQT74CDX6NC | 96.52
20191103 | nullam.lobortis@outlook.ca | 3 | Kirestin Contreras | EFB51OVO5JX | 96.62
20191104 | velit.cras.lorem@icloud.ca | 1 | Hayden Mccray | WSX15GTI5EH | 97.85
20191105 | volutpat.nulla.dignissim@outlook.org | 1 | Barclay Park | ATX18HPX5EG | 97.71
20191106 | odio.tristique@yahoo.com | 4 | Joshua Waters | EUQ17BXU3UZ | 96.98
20191107 | in.nec.orci@aol.edu | 3 | Shana Fitzgerald | TGO63OXT4MJ | 97.67
20191108 | id.blandit@outlook.com | 2 | Rudyard Wilkins | SUC56XYY6WB | 97.49

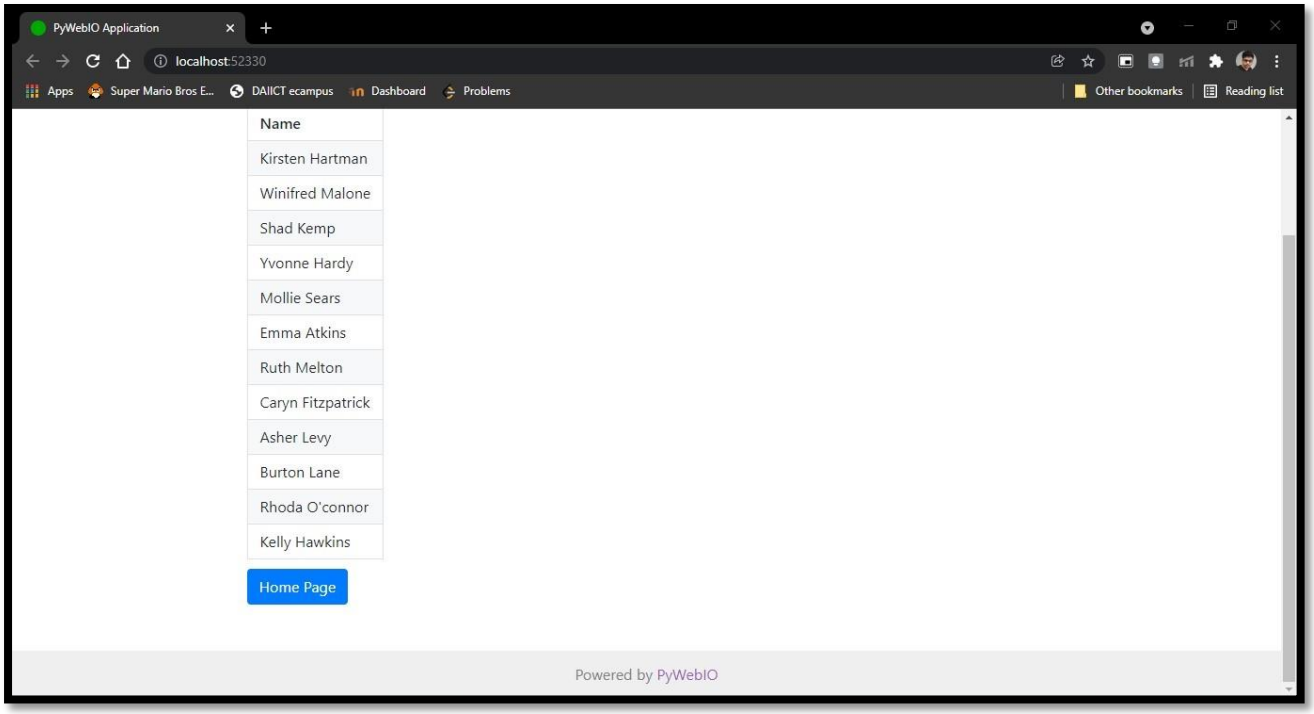

PyWebIO Application
localhost:52330
Apps
Super Mario Bros E...
DAIICT ecampus
Dashboard
Problems
Other bookmarks
Reading list
20191104 | velit.cras.lorem@icloud.ca | 1 | Hayden Mccray | WSX15GTI5EH | 97.85
20191105 | volutpat.nulla.dignissim@outlook.org | 1 | Barclay Park | ATX18HPX5EG | 97.71
20191106 | odio.tristique@yahoo.com | 4 | Joshua Waters | EUQ17BXU3UZ | 96.98
20191107 | in.nec.orci@aol.edu | 3 | Shana Fitzgerald | TGO63OXT4MJ | 97.67
20191108 | id.blandit@outlook.com | 2 | Rudyard Wilkins | SUC56XYY6WB | 97.49
20191109 | nec.quam@yahoo.couk | 2 | Garth Bradford | UHH12COD4VJ | 96.73
20191110 | elementum.lorem@yahoo.com | 1 | Fleur Hodge | XNB51LMY4YR | 97.27
Home Page
Powered by PyWebIO

Queries:

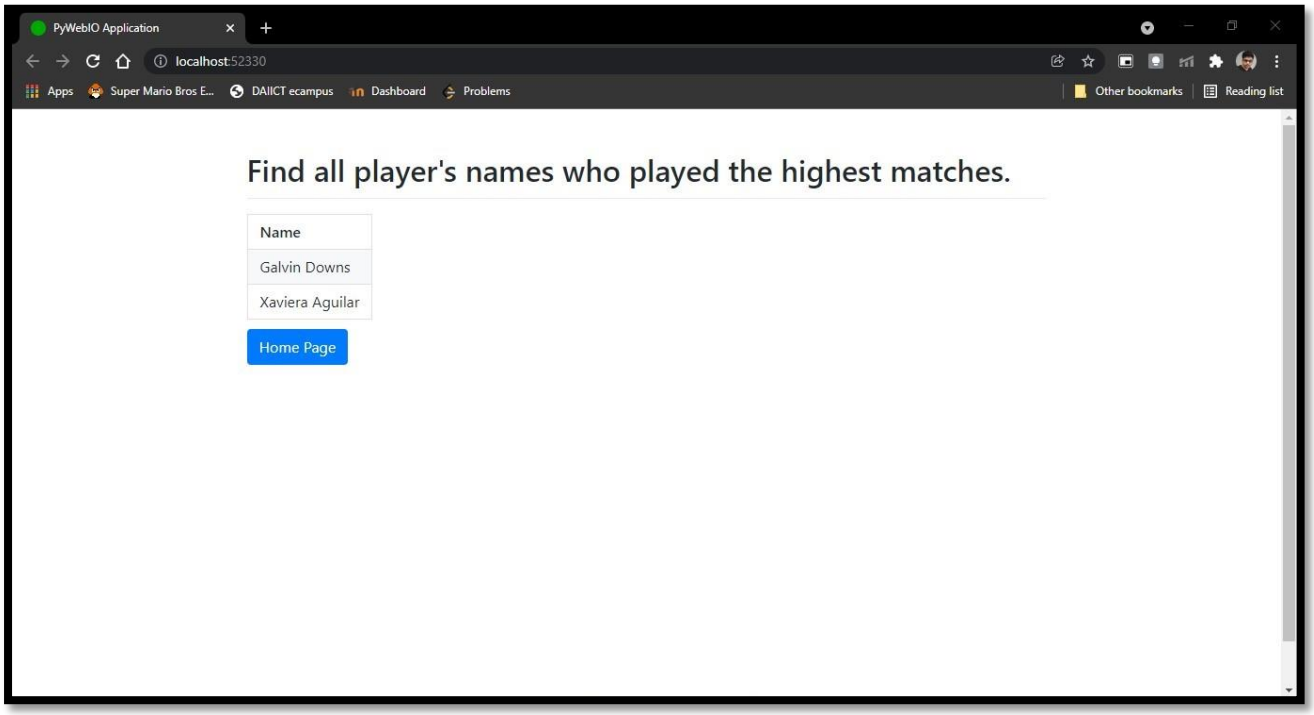
Query 1:

Query 2:

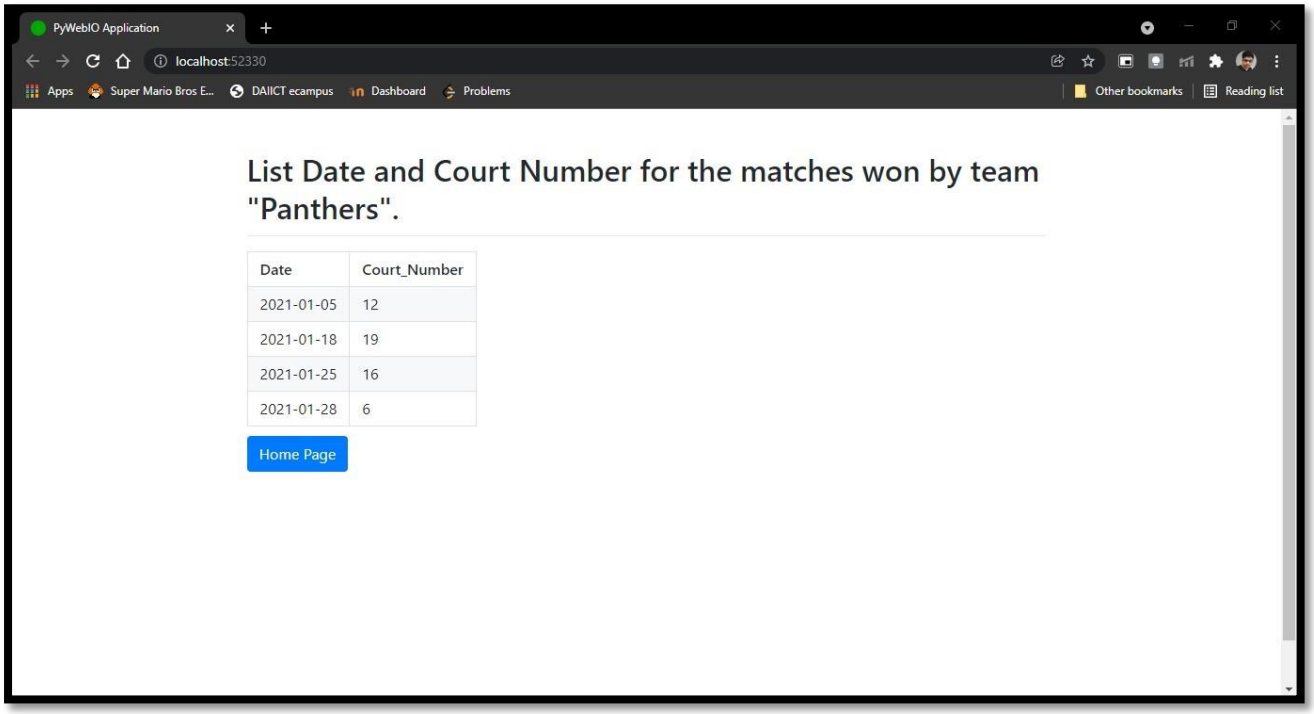

PyWebIO Application
localhost:52330
Apps
Super Mario Bros E...
DAIICT ecampus
Dashboard
Problems
Other bookmarks
Reading list
List all the player names of team who has scored highest points among other teams.
Name
Kirsten Hartman
Winifred Malone
Shad Kemp
Yvonne Hardy
Mollie Sears
Emma Atkins
Ruth Melton
Caryn Fitzpatrick
Asher Levy
Burton Lane
Rhoda O'connor
Kelly Hawkins


PyWebIO Application
localhost:52330
Apps
Super Mario Bros E...
DAIICT ecampus
Dashboard
Problems
Other bookmarks
Reading list
Name
Kirsten Hartman
Winifred Malone
Shad Kemp
Yvonne Hardy
Mollie Sears
Emma Atkins
Ruth Melton
Caryn Fitzpatrick
Asher Levy
Burton Lane
Rhoda O'connor
Kelly Hawkins
Home Page
Powered by PyWebIO

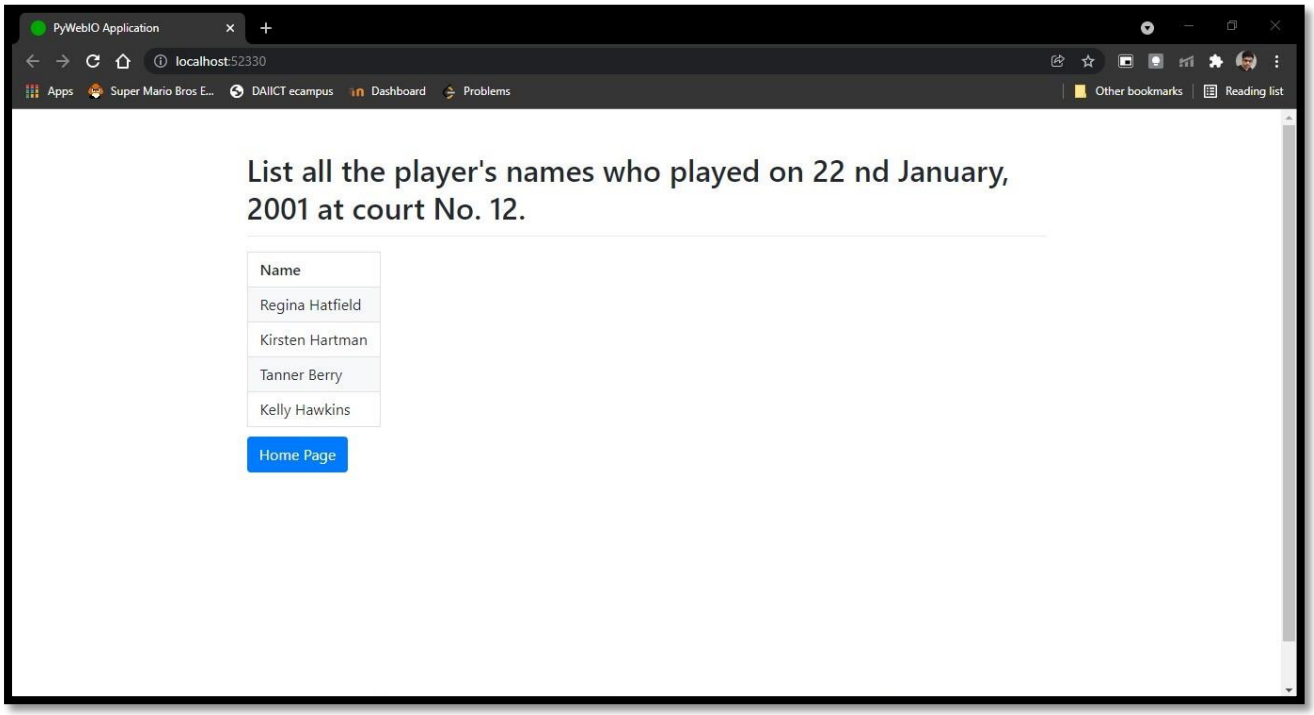
Query 3:



Query 4:

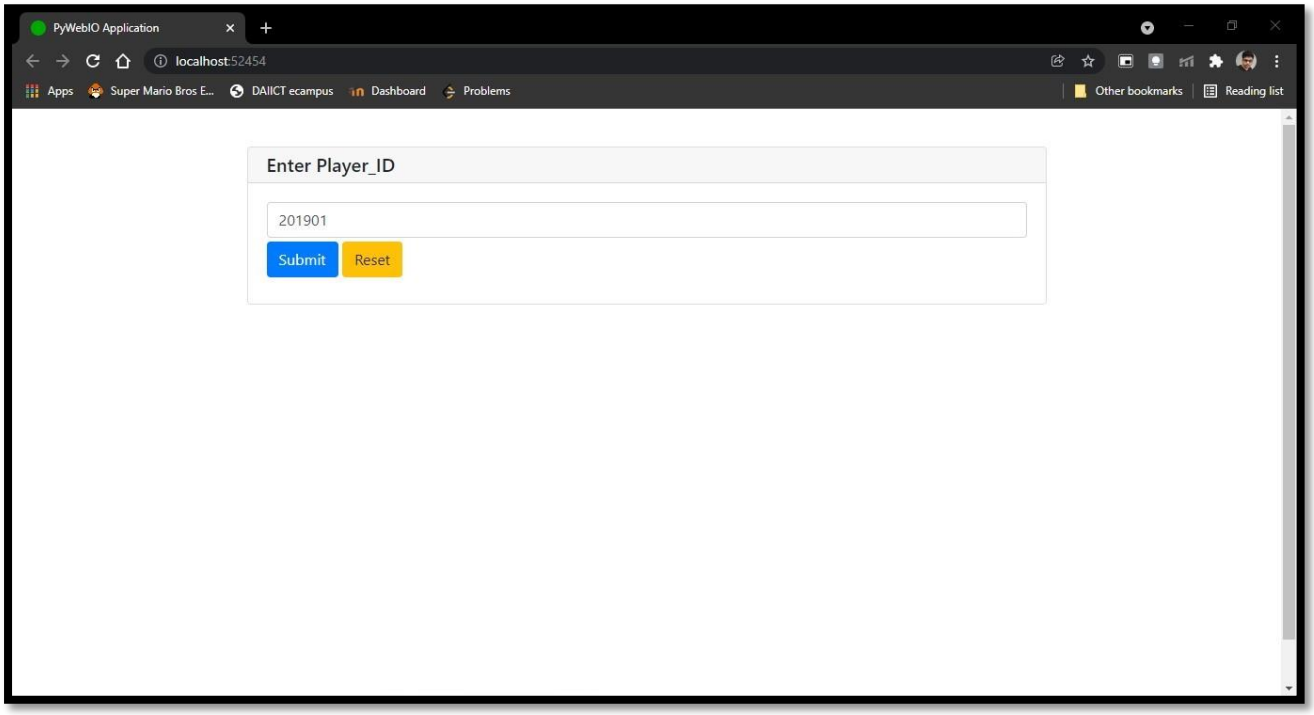


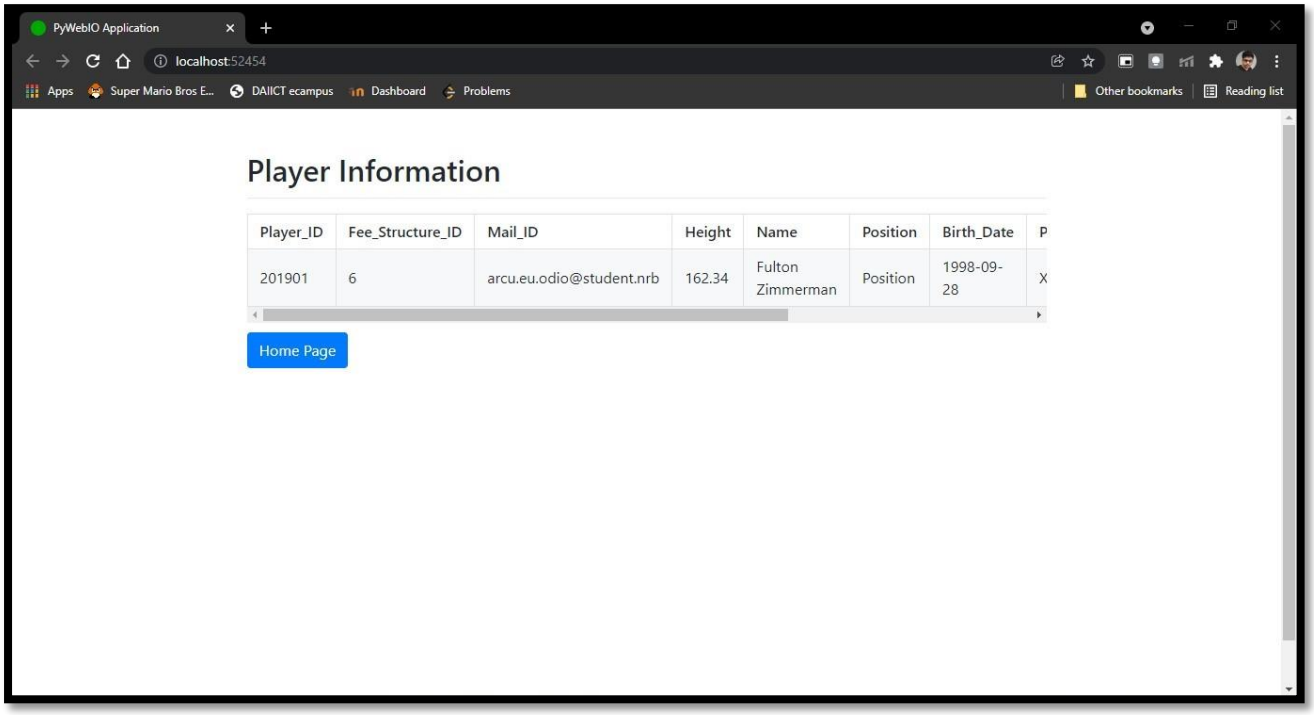
Query 5:


PyWebIO Application
localhost:52330
Apps
Super Mario Bros E...
DAIICT ecampus
Dashboard
Problems
Other bookmarks
Reading list
List all the player's names who played on 22 nd January, 2001 at court No. 12.
Name
Regina Hatfield
Kirsten Hartman
Tanner Berry
Kelly Hawkins
Home Page

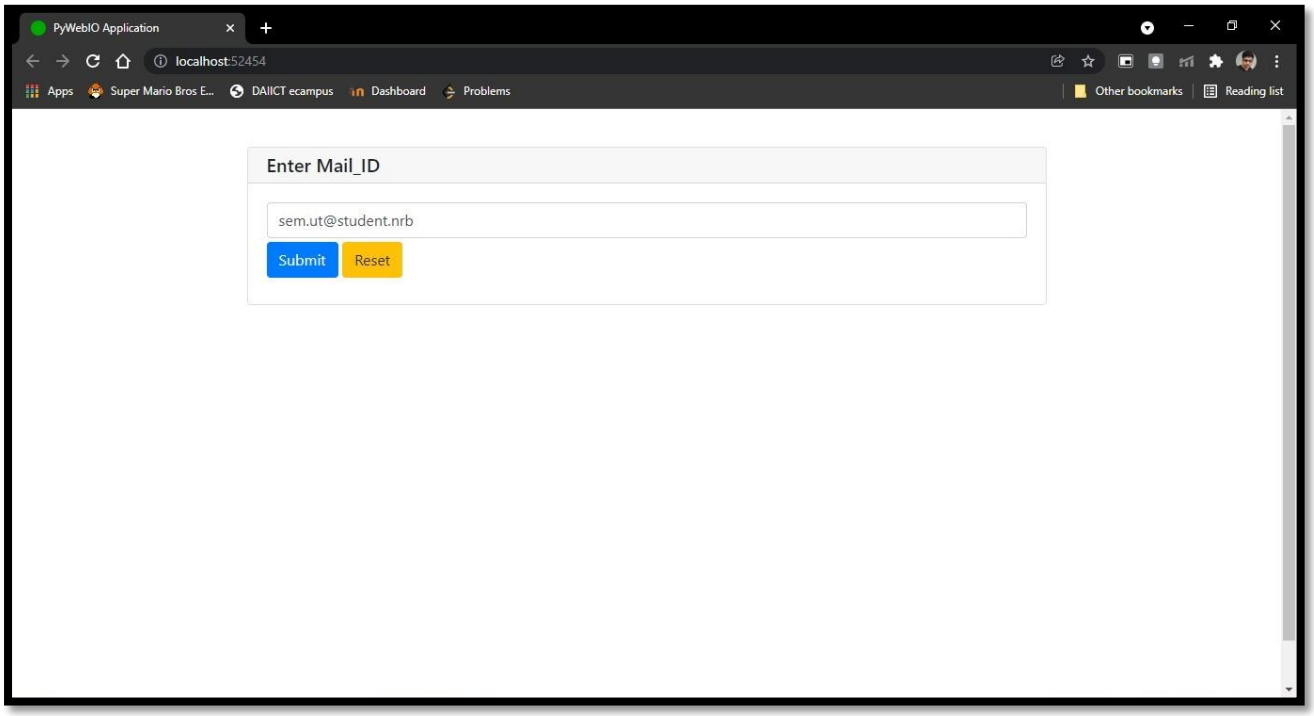
Search:

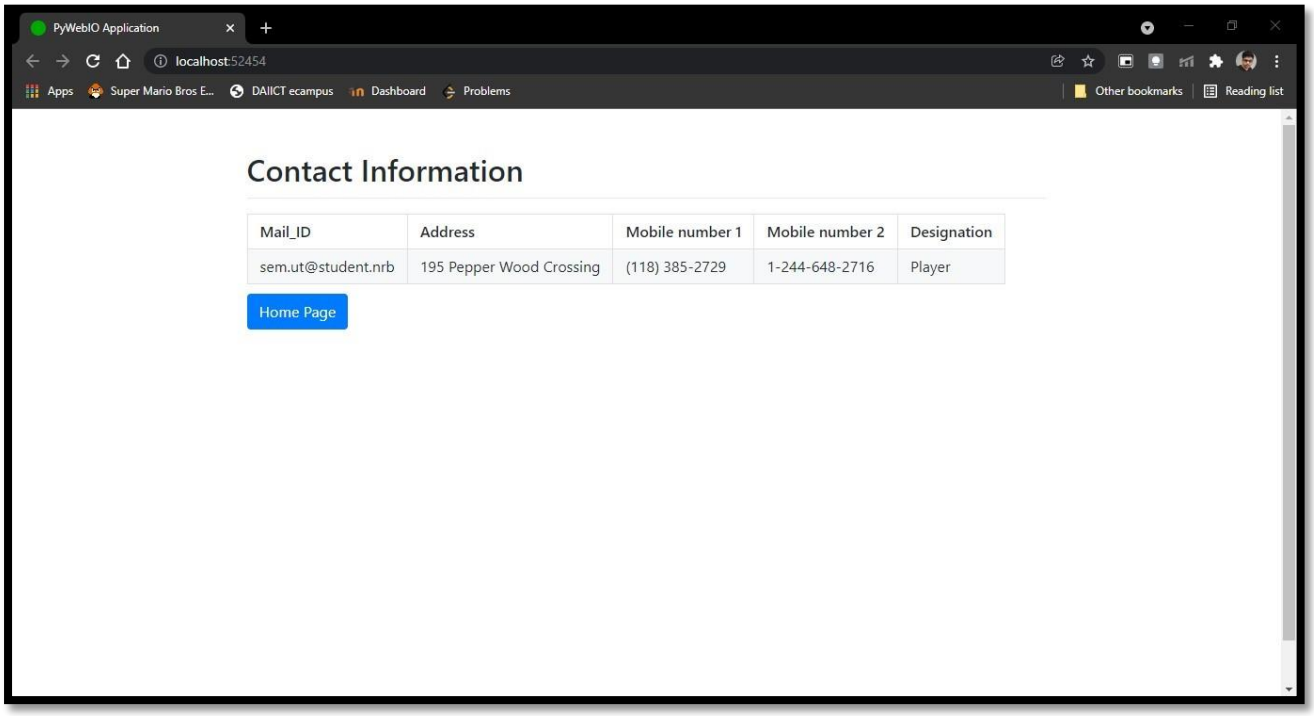
Search Player:





Search Contact Information:



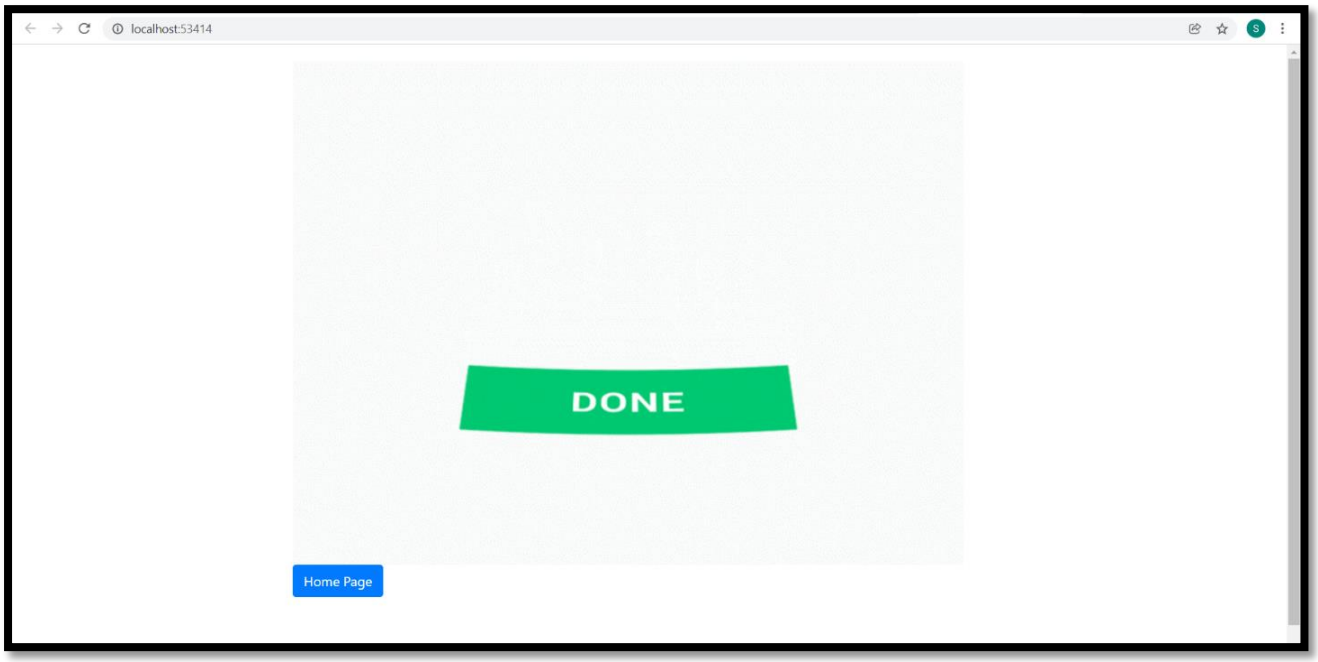


Insert and Delete Demo using GUI:

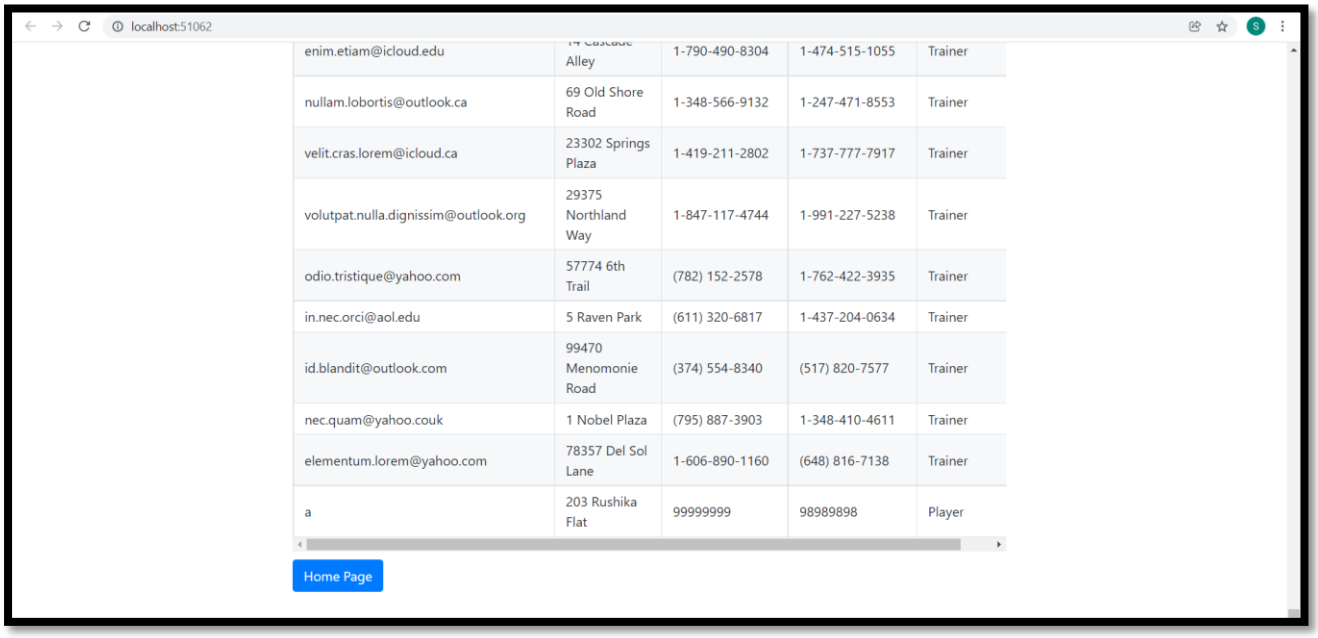
For Contact Information Table:

Insert Form:

After Clicking Submit:

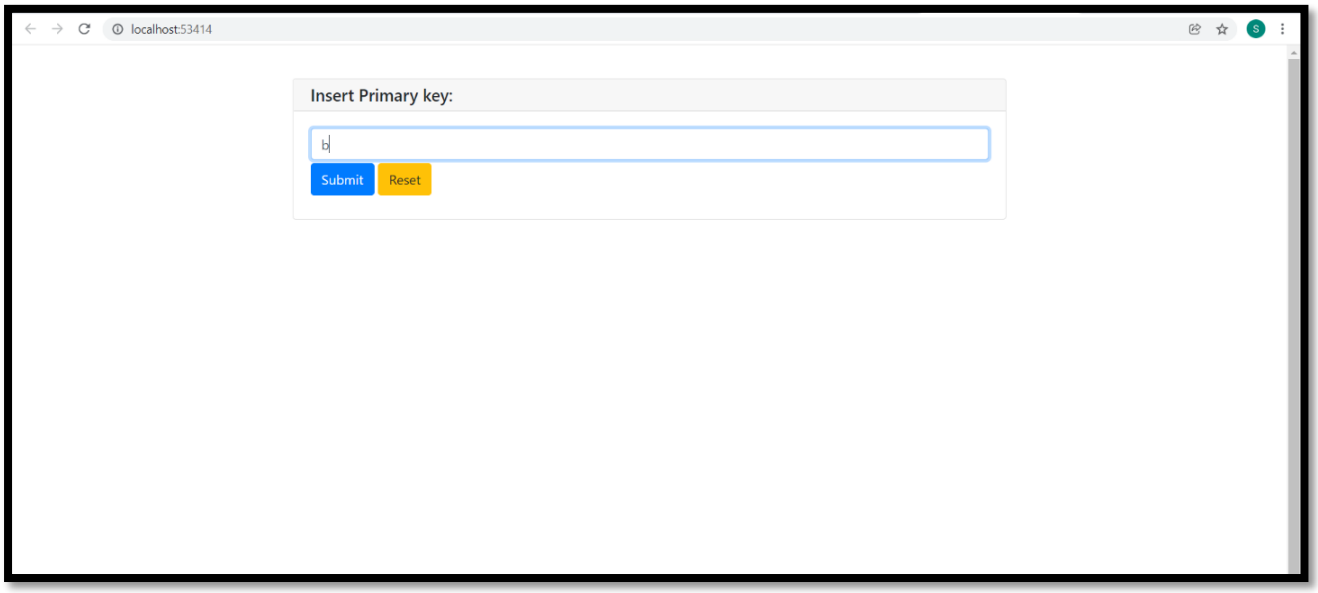


Verification:

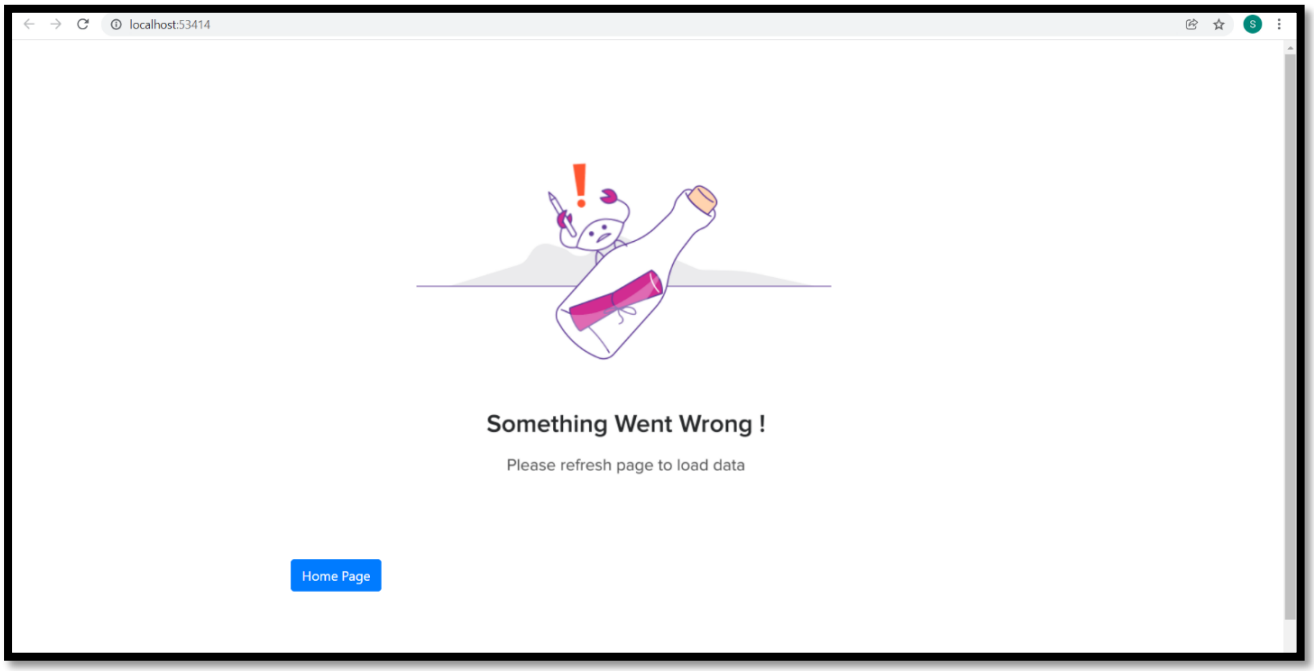


Delete:

Case 1: If user enters non-existing primary key by mistake,

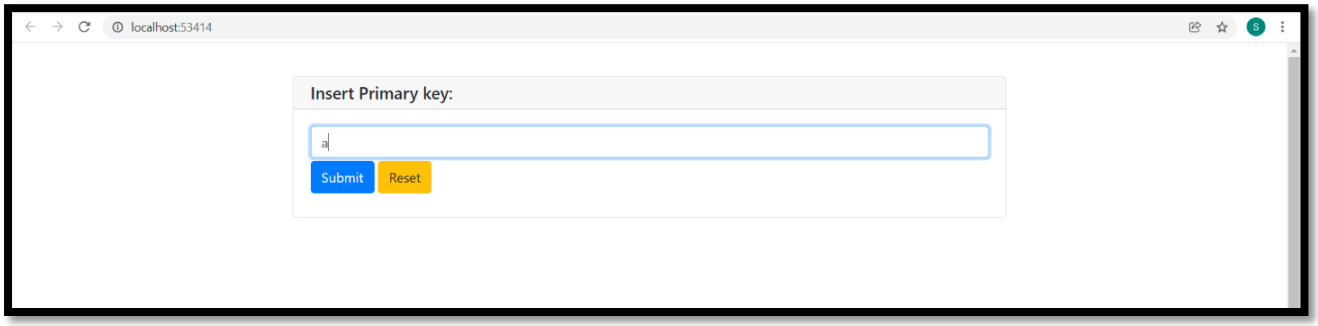


After Clicking Submit:

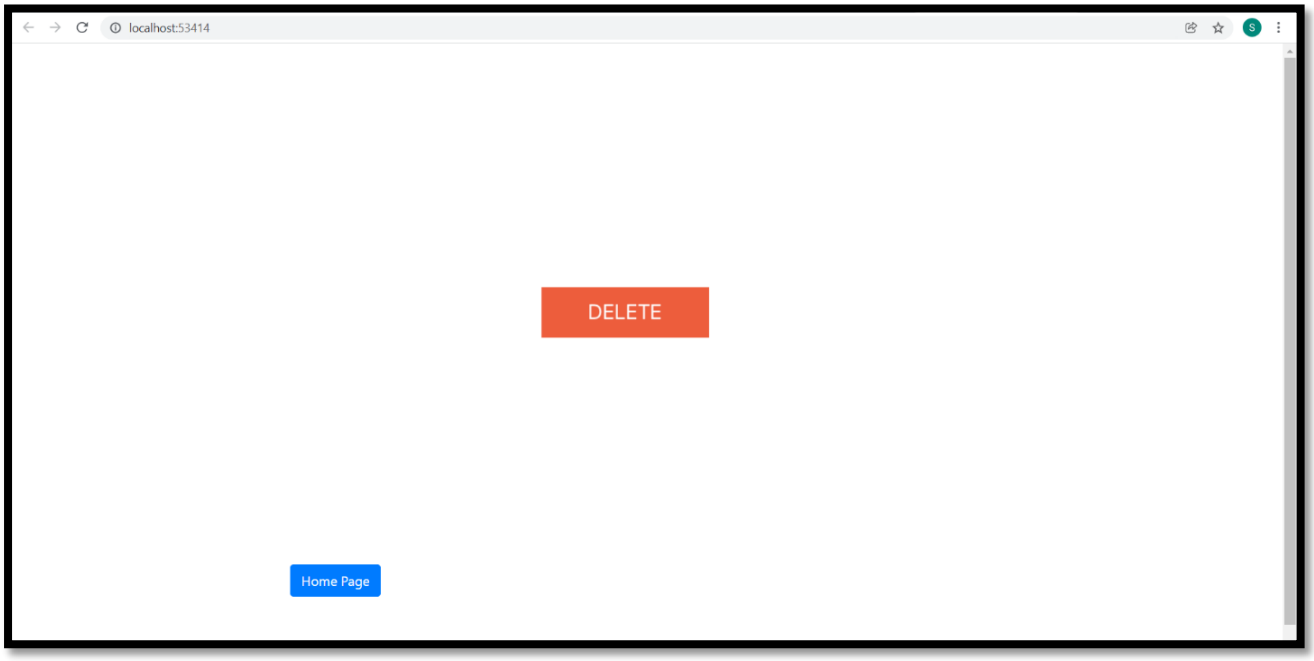


Case 2:

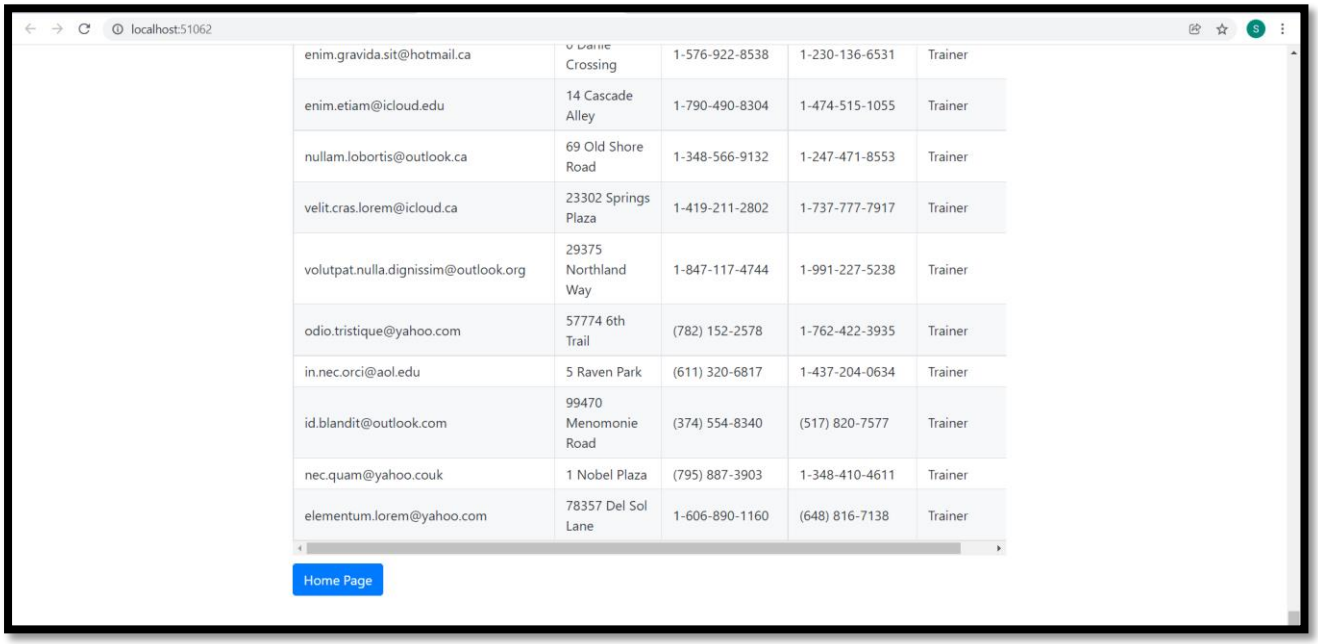
Existing Primary key is entered:



After clicking submit:

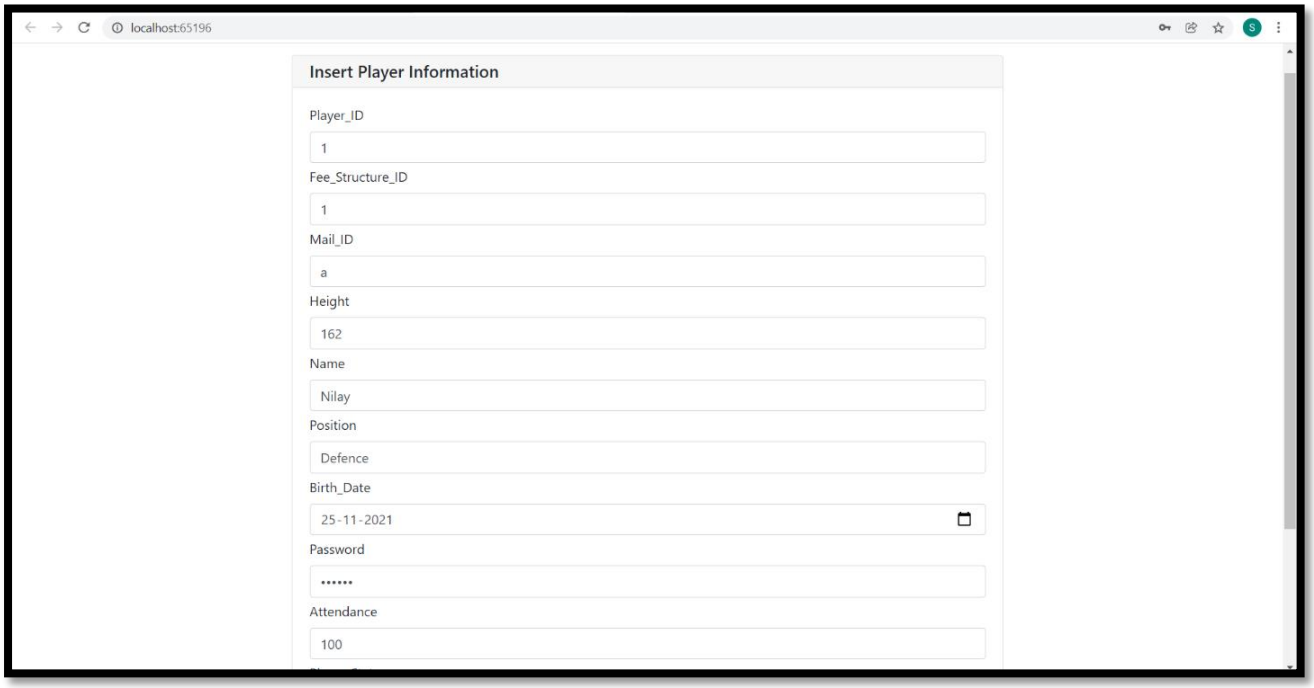


Verification:

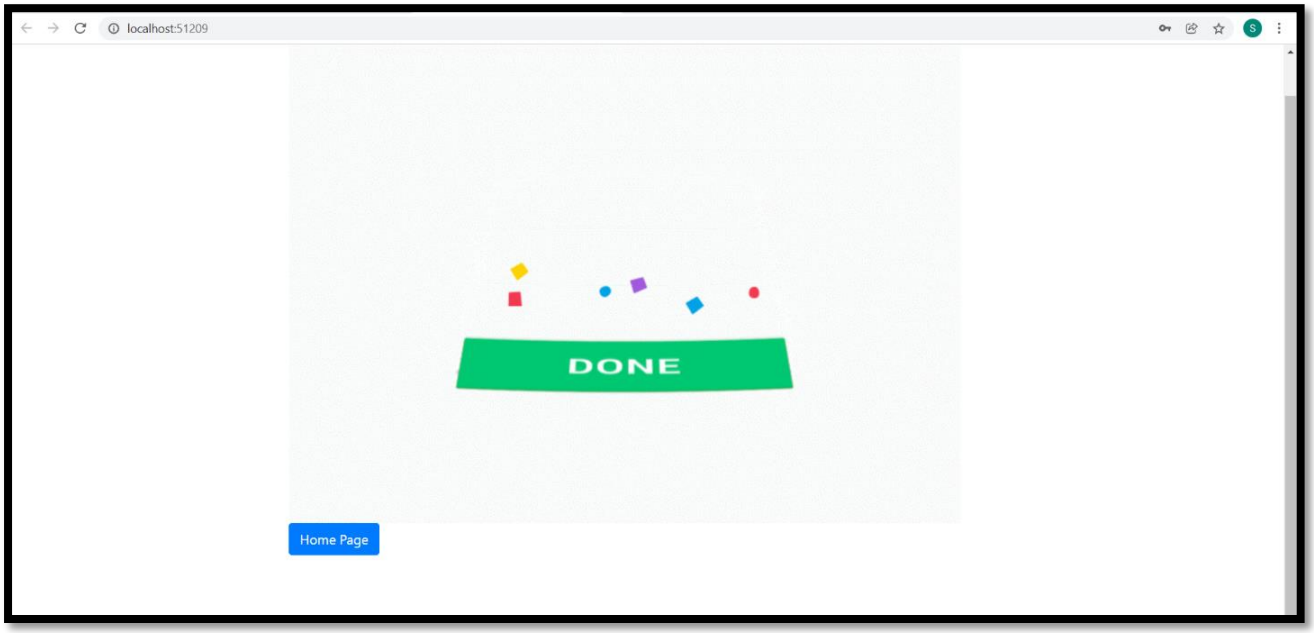


Player:

Insert Form:

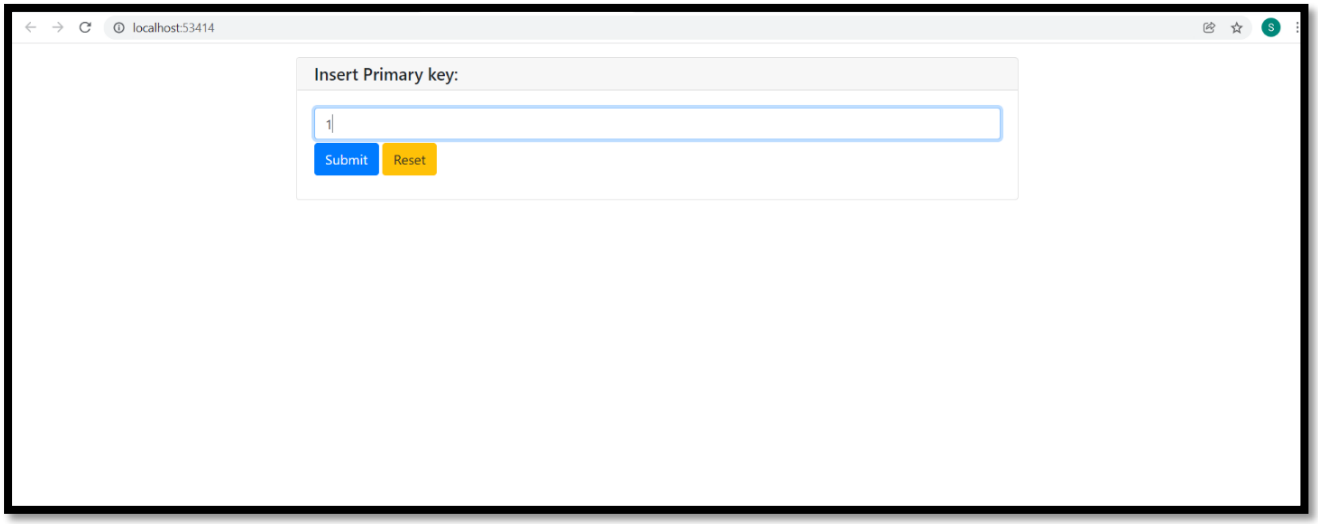


After Clicking Submit:

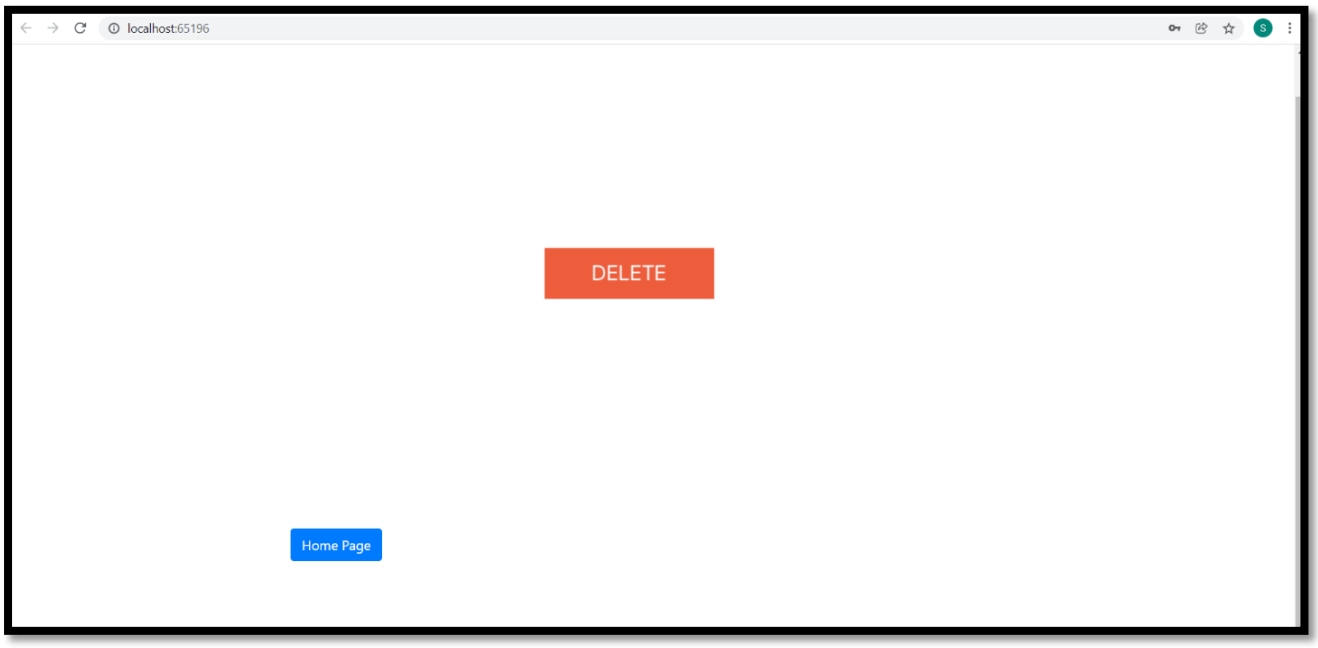


Delete:

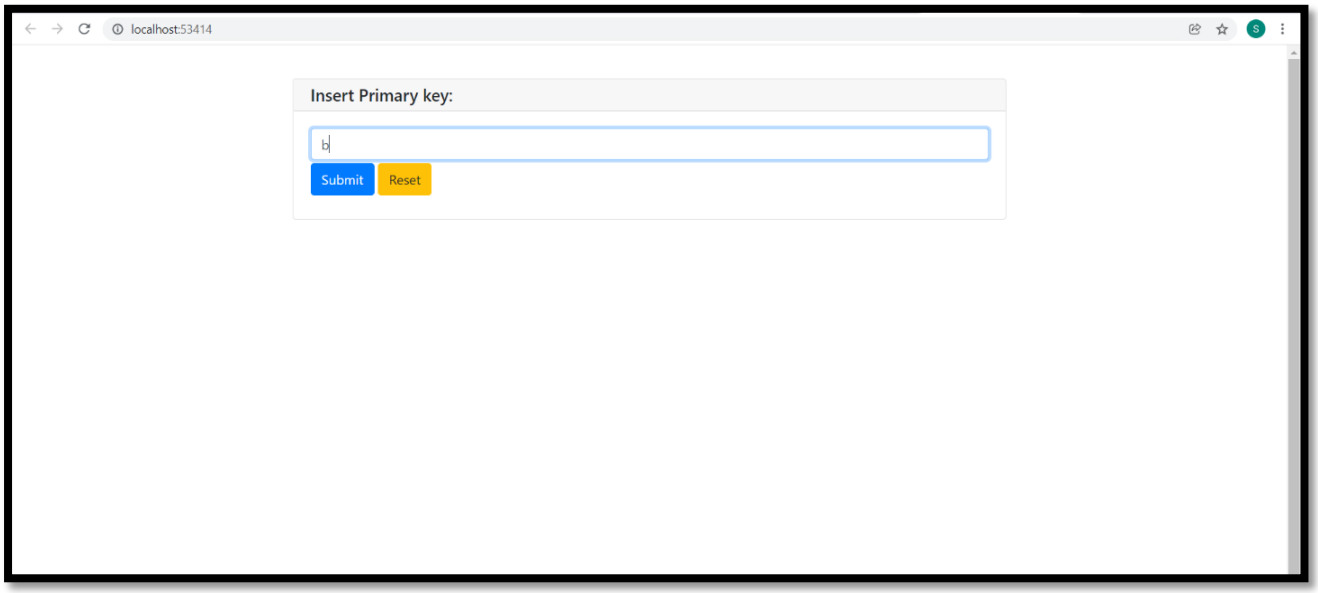
Case 1: If data is existing.



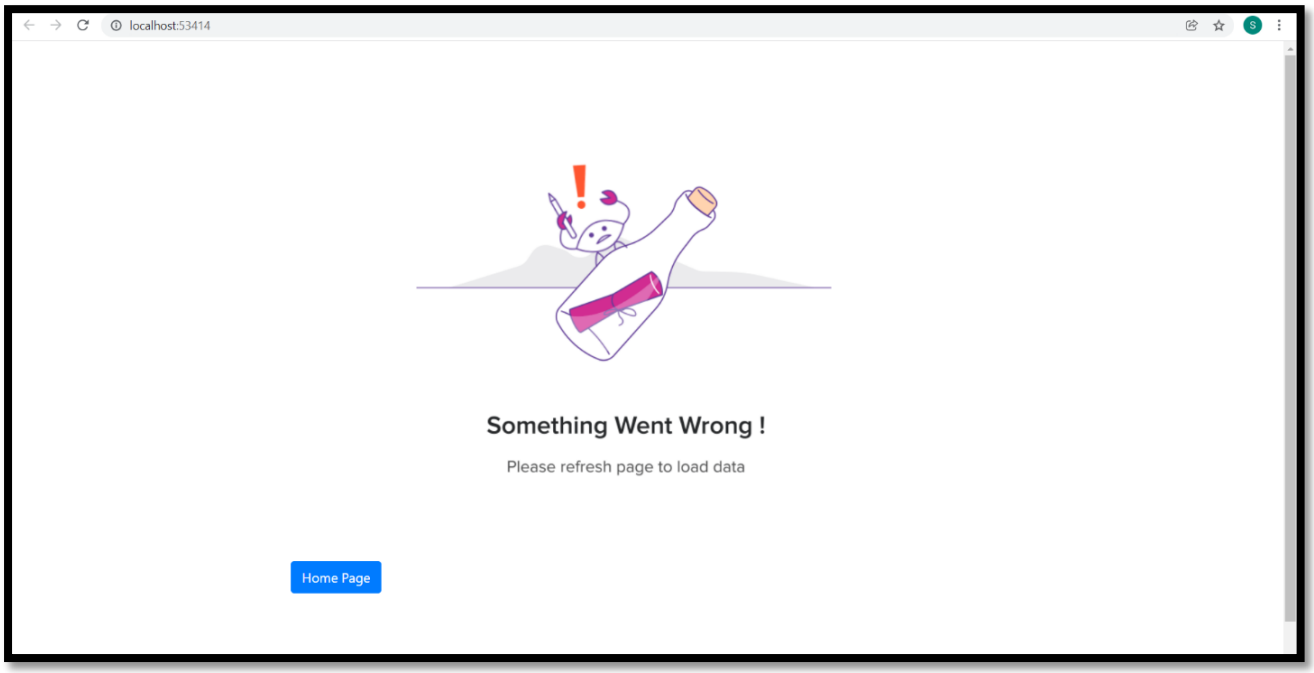
After clicking submit:



Case 2: If data is not existed:

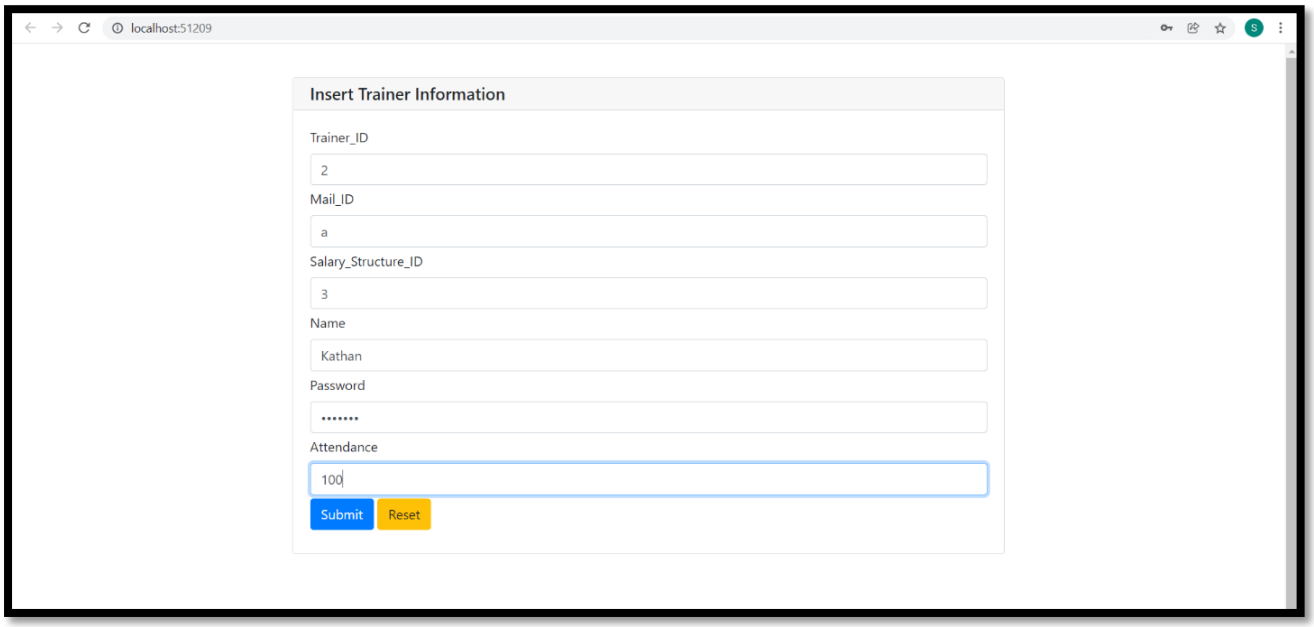


After Clicking Submit:

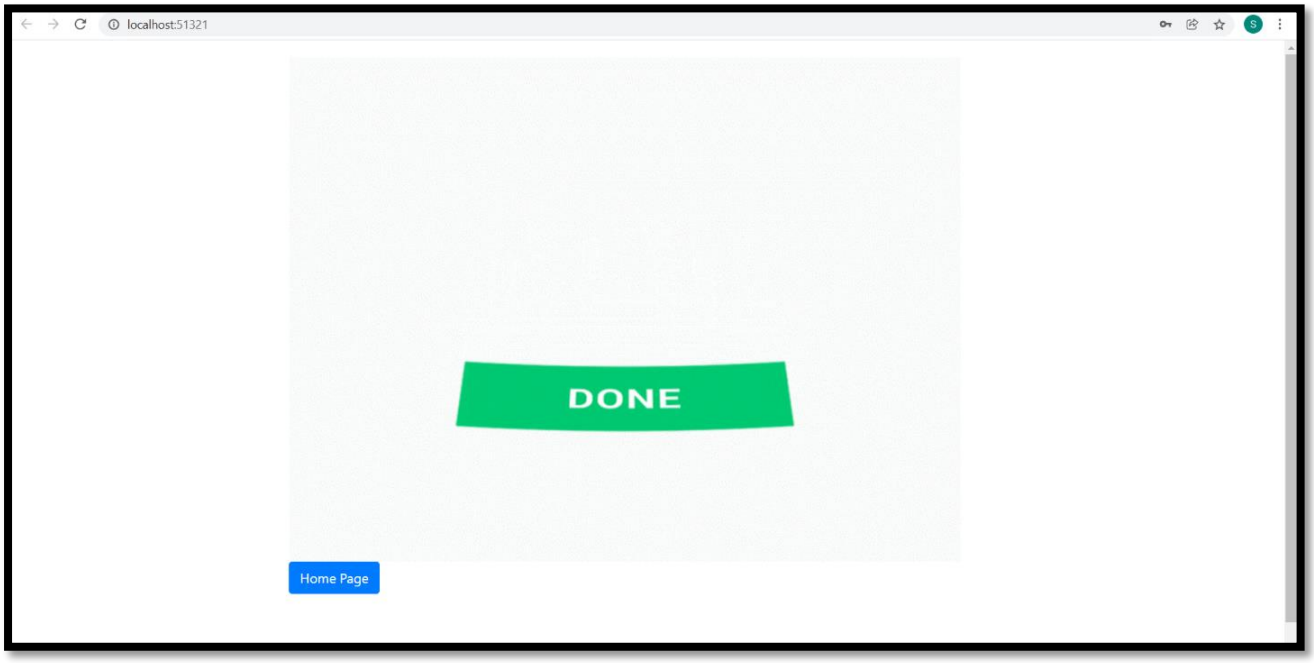


Trainer:

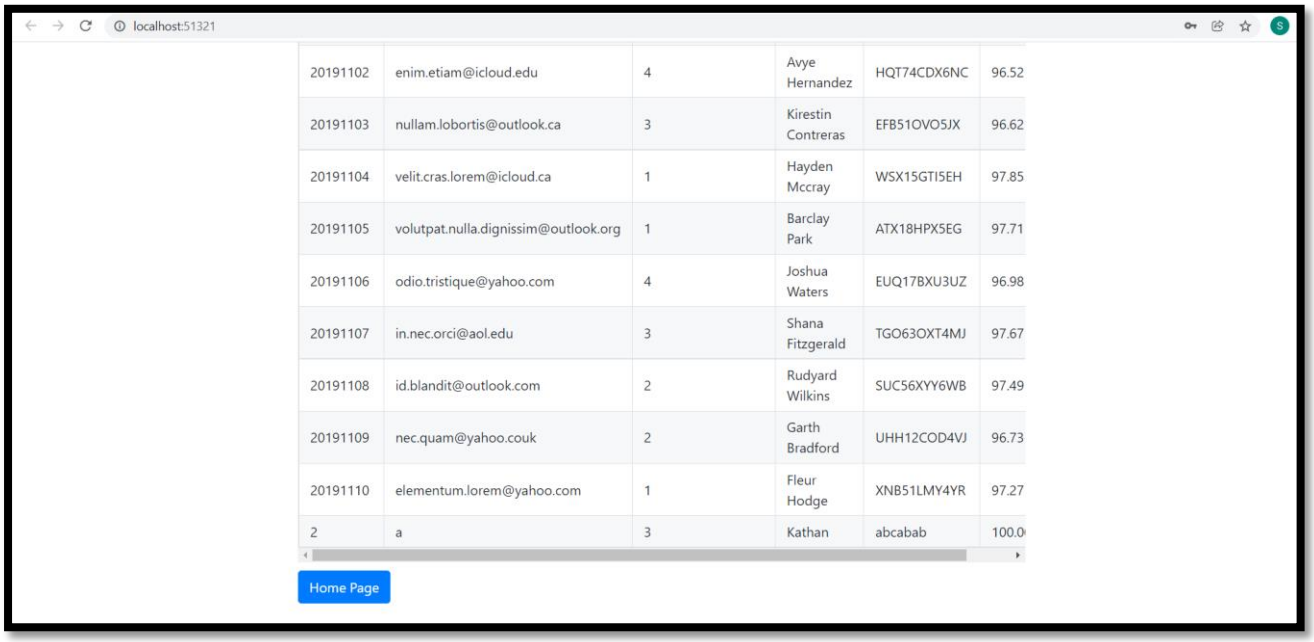
Insert Form:



After Clicking Submit:

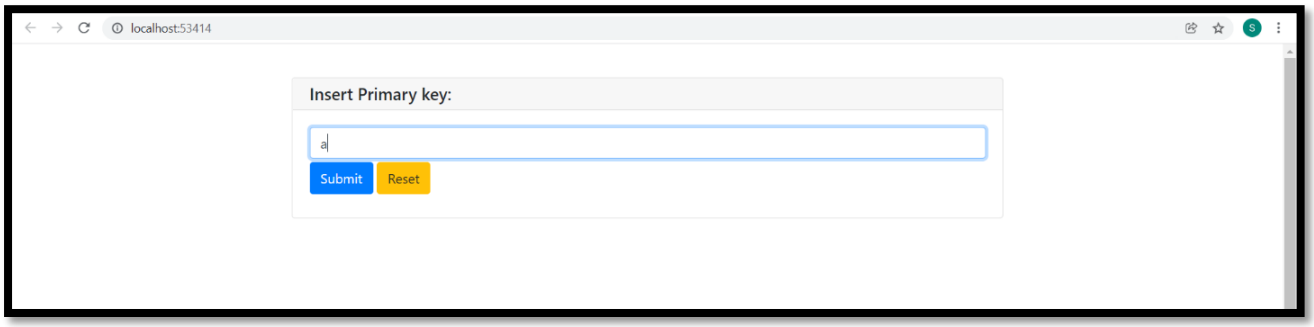


Verification:

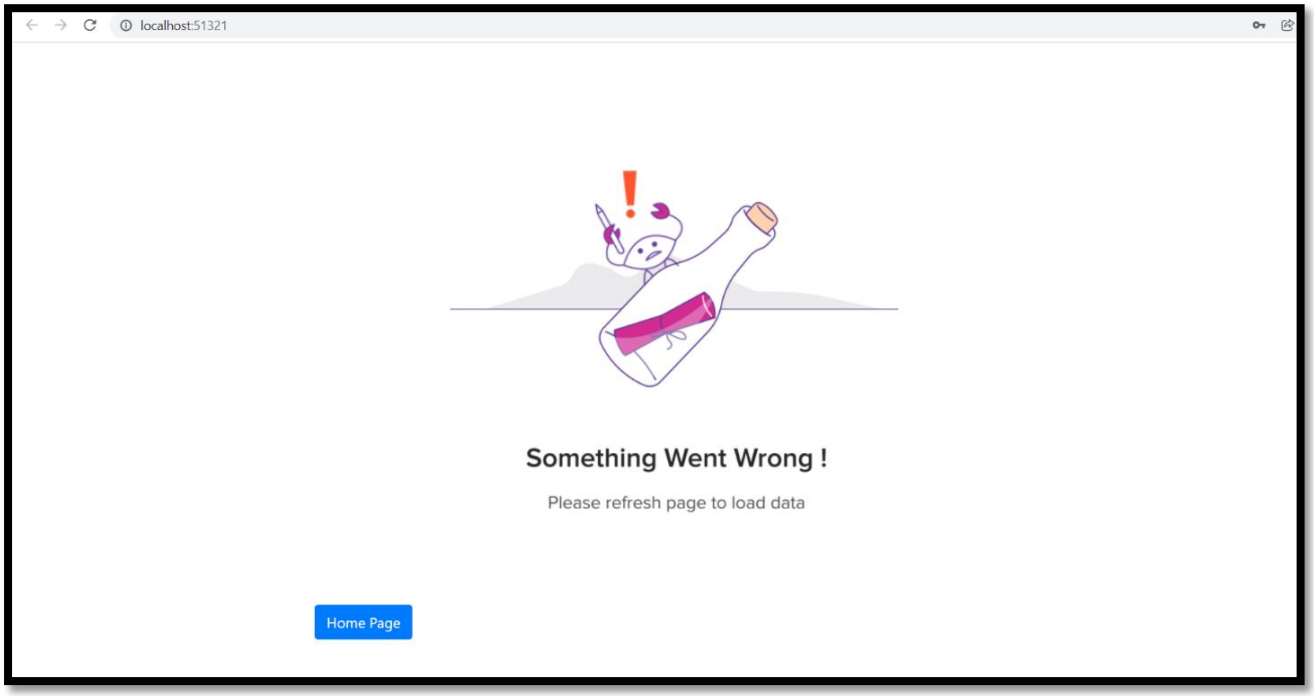


Delete:

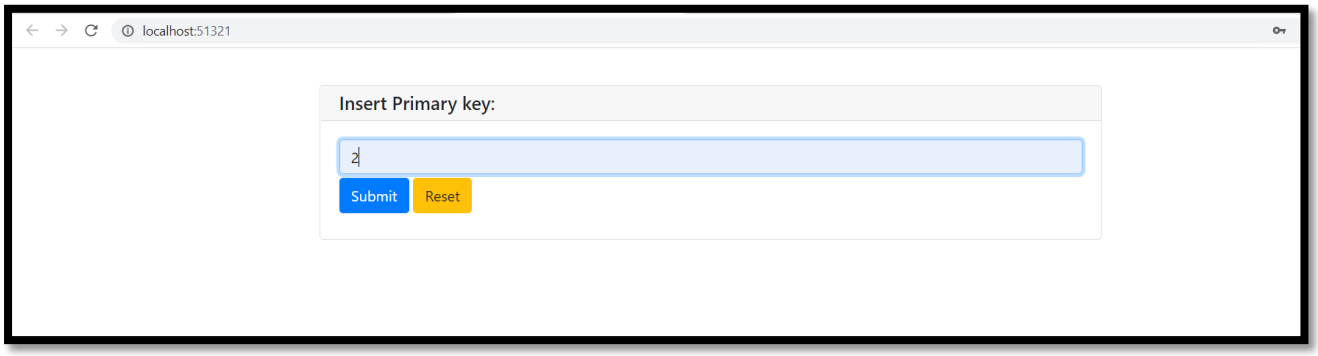
Case 1: If data does not exist:



Result:



Case 2: If data is already existing



Result: