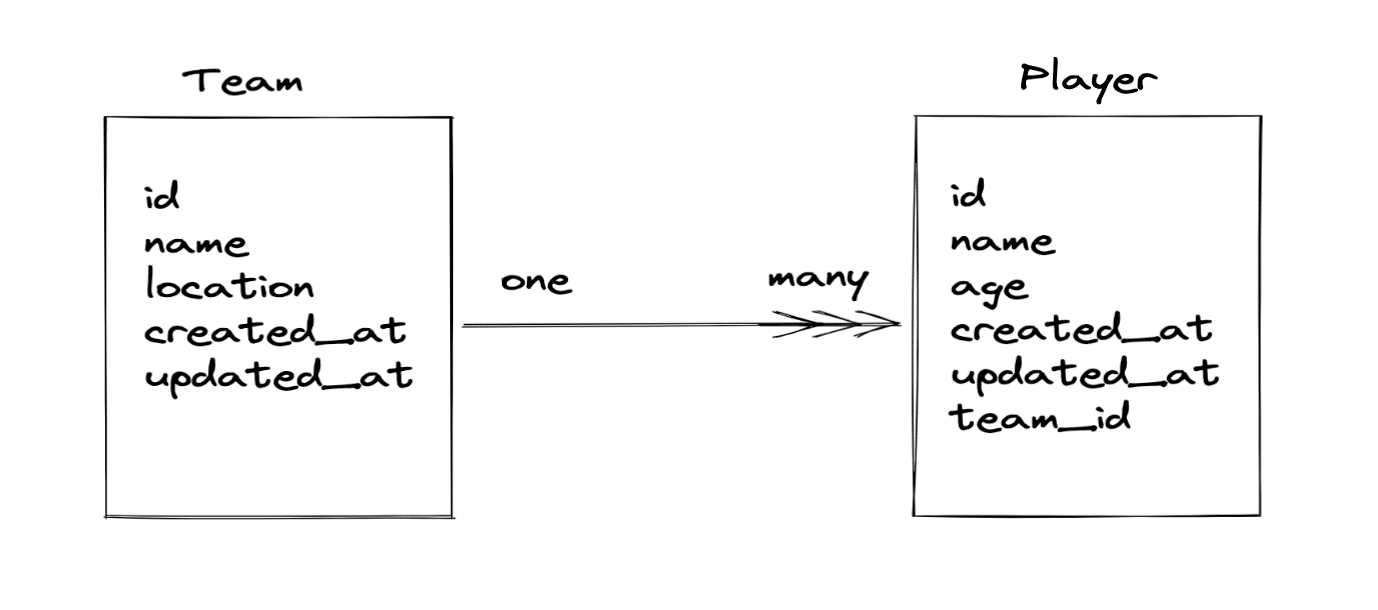
Assignment 2: Java Backend

**REST API's using Spring Boot for two models Team and Player.**

I have created REST APIs for two models i.e., Team and Player using Spring Boot to perform CRUD operation. These models have one to many relationships and below is the ER diagram for the same.



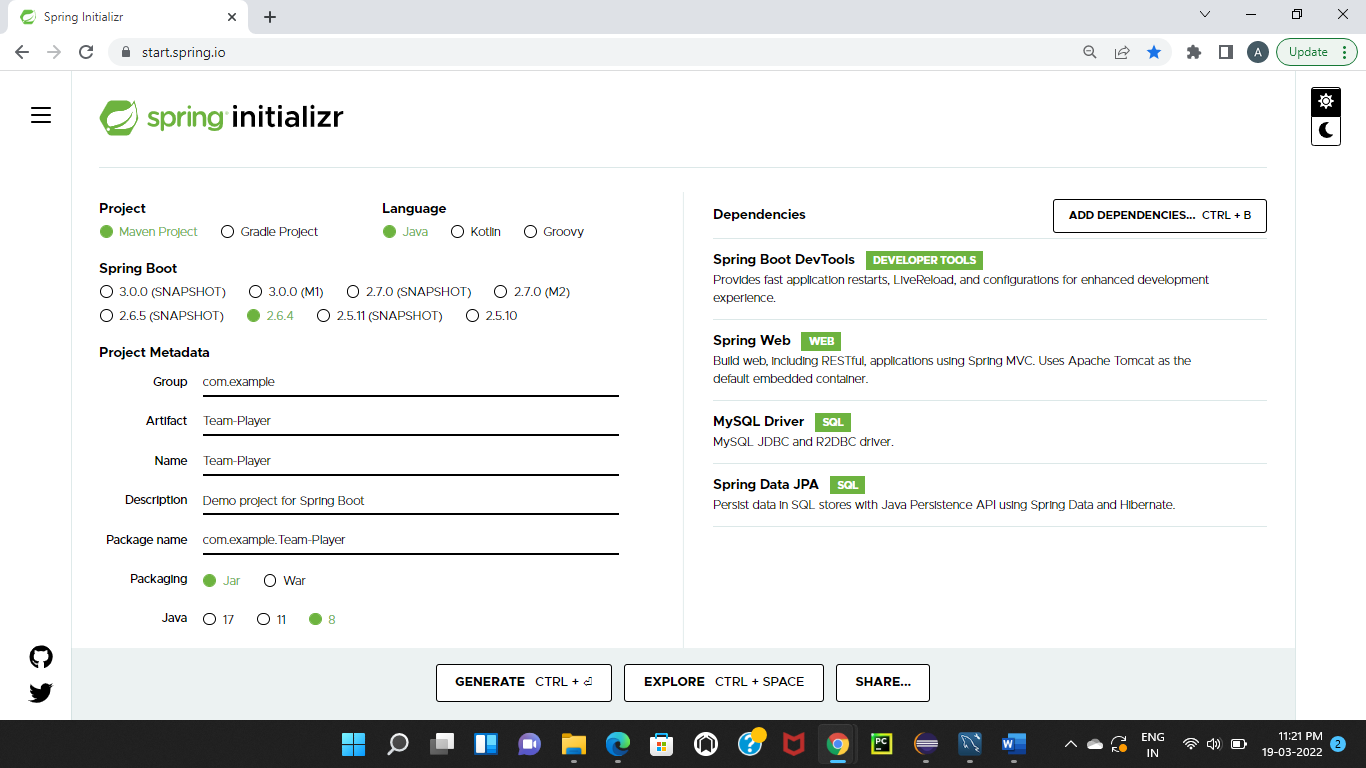
**Tool/Platform used**

* Java
* Spring Boot Framework
* Maven
* MySQL
* Eclipse IDE
* Postman

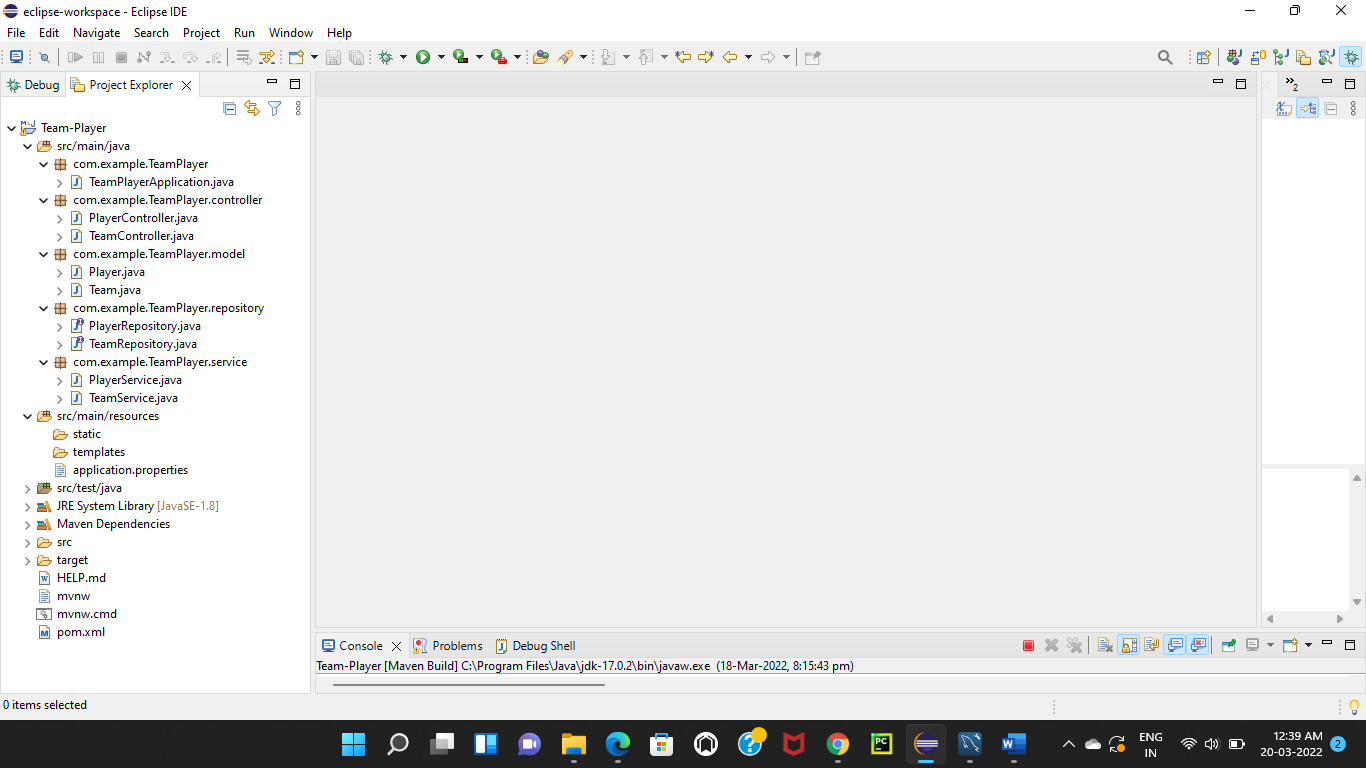
To create above spring boot project, I have used spring initializr tool which provides basic structure of spring boot project to get started.

For this project, I have added following dependencies:

* **Spring boot DevTools** – Provides Development Tools
* **Spring Web** – For building restful web application
* **MySQL Driver** – For JDBC Driver
* **Spring data JPA** – Java persistence API and Hibernate which requires to access the data from database.



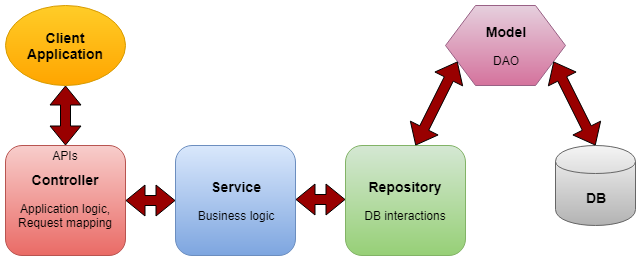
Below is Folder structure of zip the extracted folder generated from spring initializr.



In above folder, TeamPlayerApplication.java is the main file that will launch the Spring Boot Application. The application.properties file is use to configure the application. Configuration such as database URL, username, password. Pom.xml contains all the dependencies that I have added while creating the Spring Boot project in Spring Initializr. I have created sub packages for each of MVC (Model-View-Controller) layers in the project.

* **Controller** – This is top most layer which is called when a request come for a particular REST API and returns an HTTP response to the client. It acts as bridge between view and model.
* **Model** – This contains all java objects which is mapped with database table.
* **Repository** – This is similar to DAO (Data Access Object) which is use to connect to the database and accesses the data.
* **Service** – This layer calls the repository to get the data and perform logic like calculations on data received, filtering data based on some logic, etc.

**Below is the architecture of spring MVC:**



**Controller-Service-Repository Architecture**

**Following are (CRUD) endpoints in spring boot**

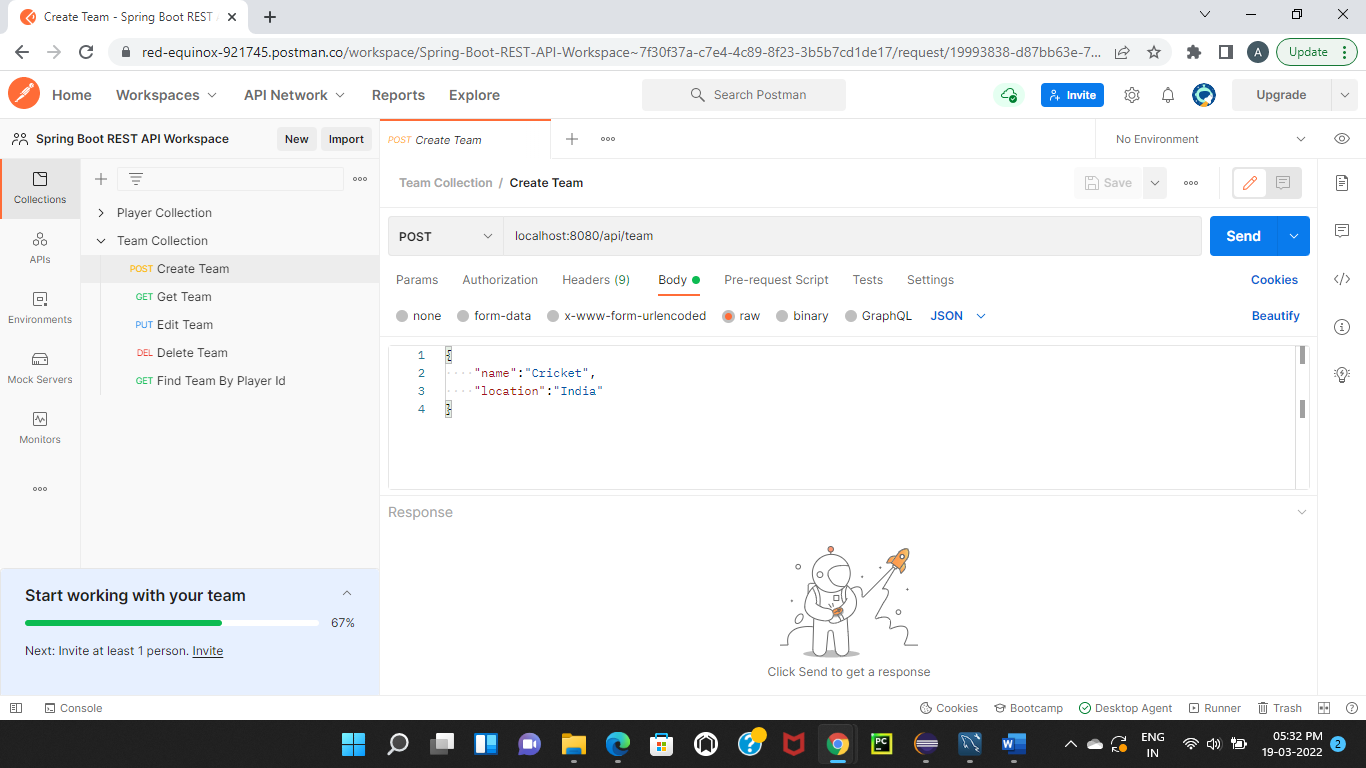
1. Create - **Team**, **Player**
2. Find by Id - **Team**, **Player**
3. update - **Team**, **Player**
4. delete - **Team**, **Player**
5. Find players by team id and Find team by player id

**Create**

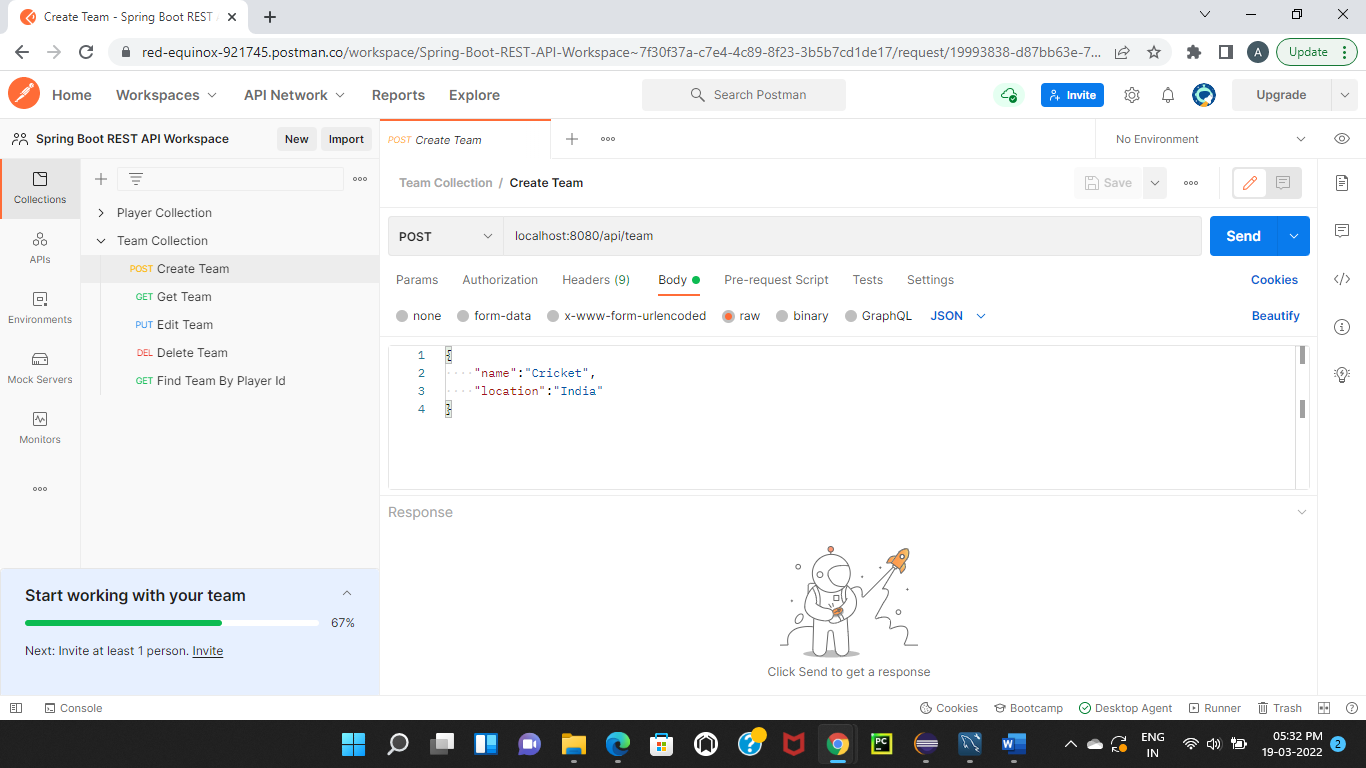
**Team**

To create a Team, I have sent POST request for the URL <https://localhost:8080/api/team>. In the URL, I have given RequestMapping as “/api/team” which is working for create endpoint and it is preceded by server and port. Here server is localhost and port are 8080.

@RequestMapping annotation in the class which comes under Controller package defines a base URL for all the REST APIs created in the controller. This base URL is followed by individual REST endpoints given to each of the controller method.

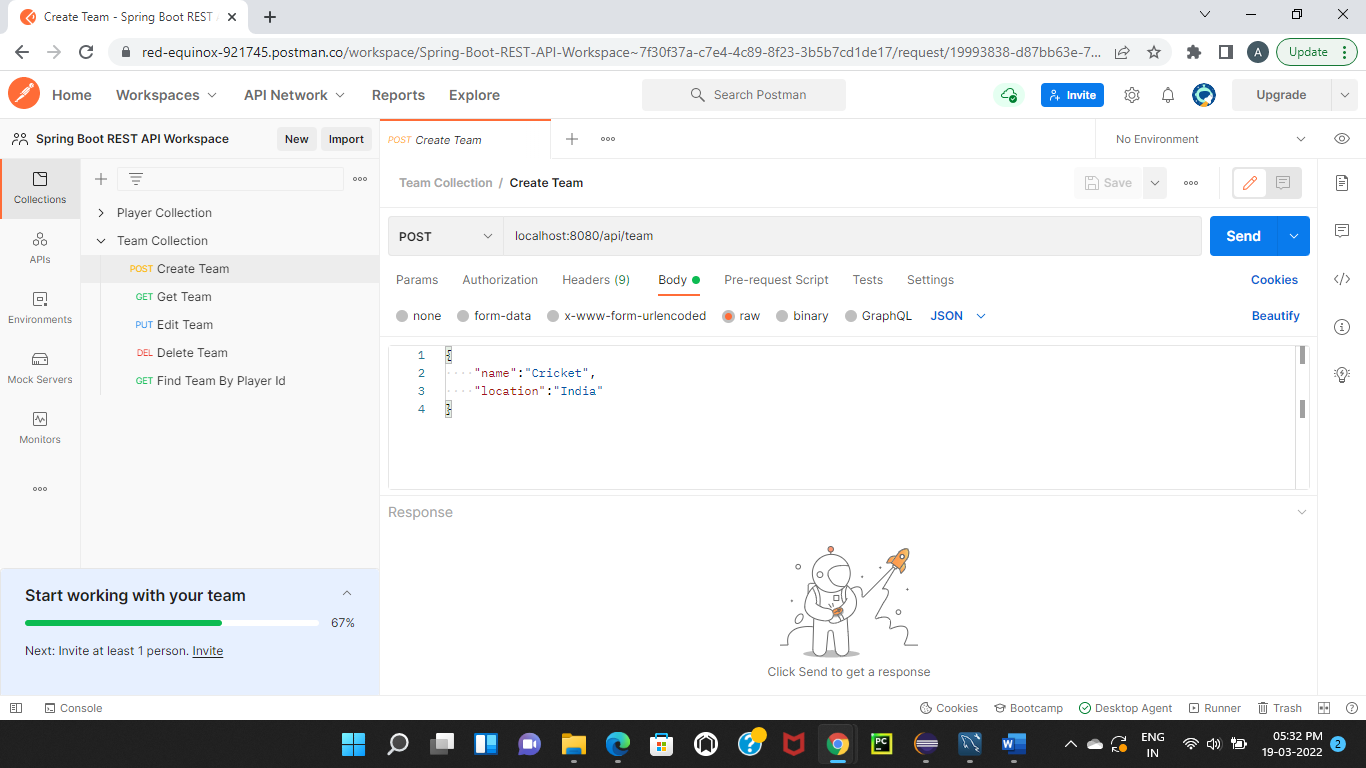


Entered below data to send request body to the given URL endpoint. These are the Team data that will be inserted into the database.

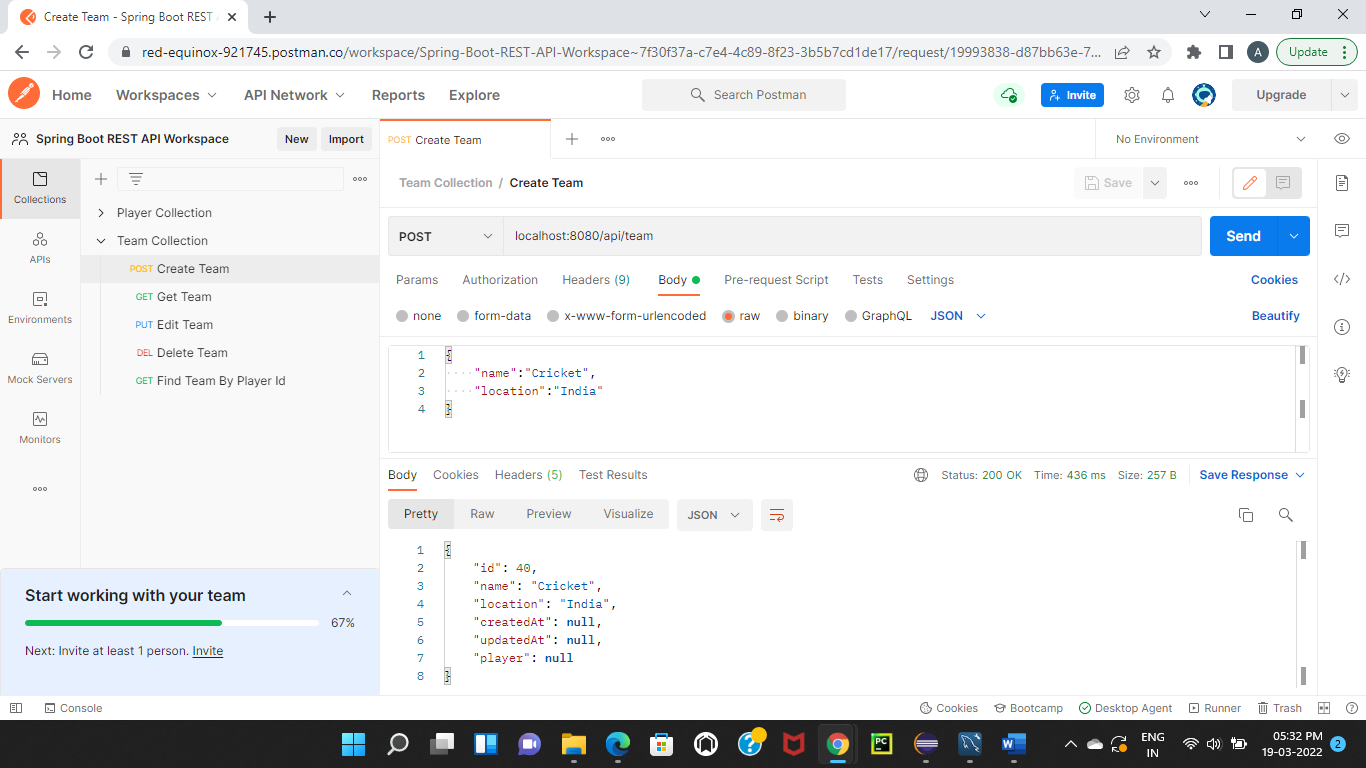


After clinking on send button it will return the response body is a JSON containing the team record details which will save in the database.

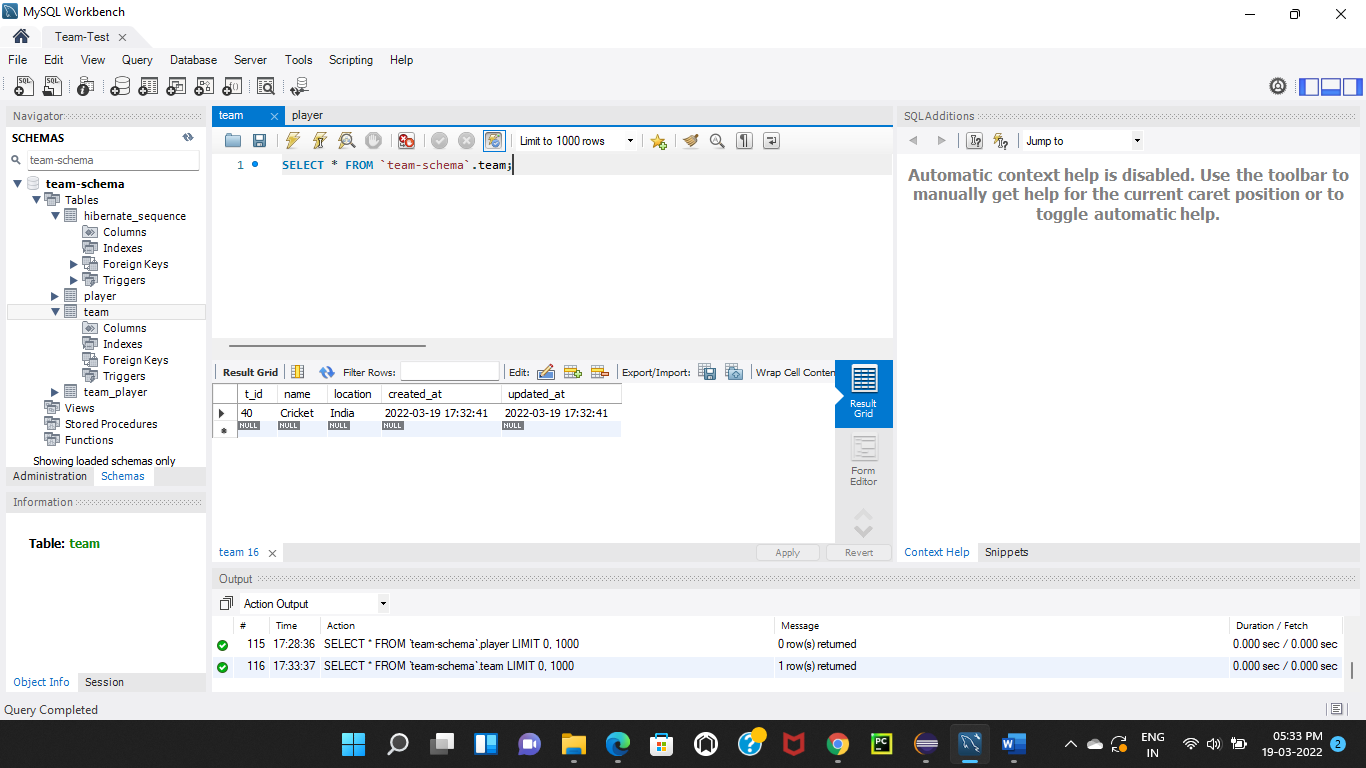
Below is the image of before sending POST request to create team.



Below is the image of after sending POST request and getting response of created team.



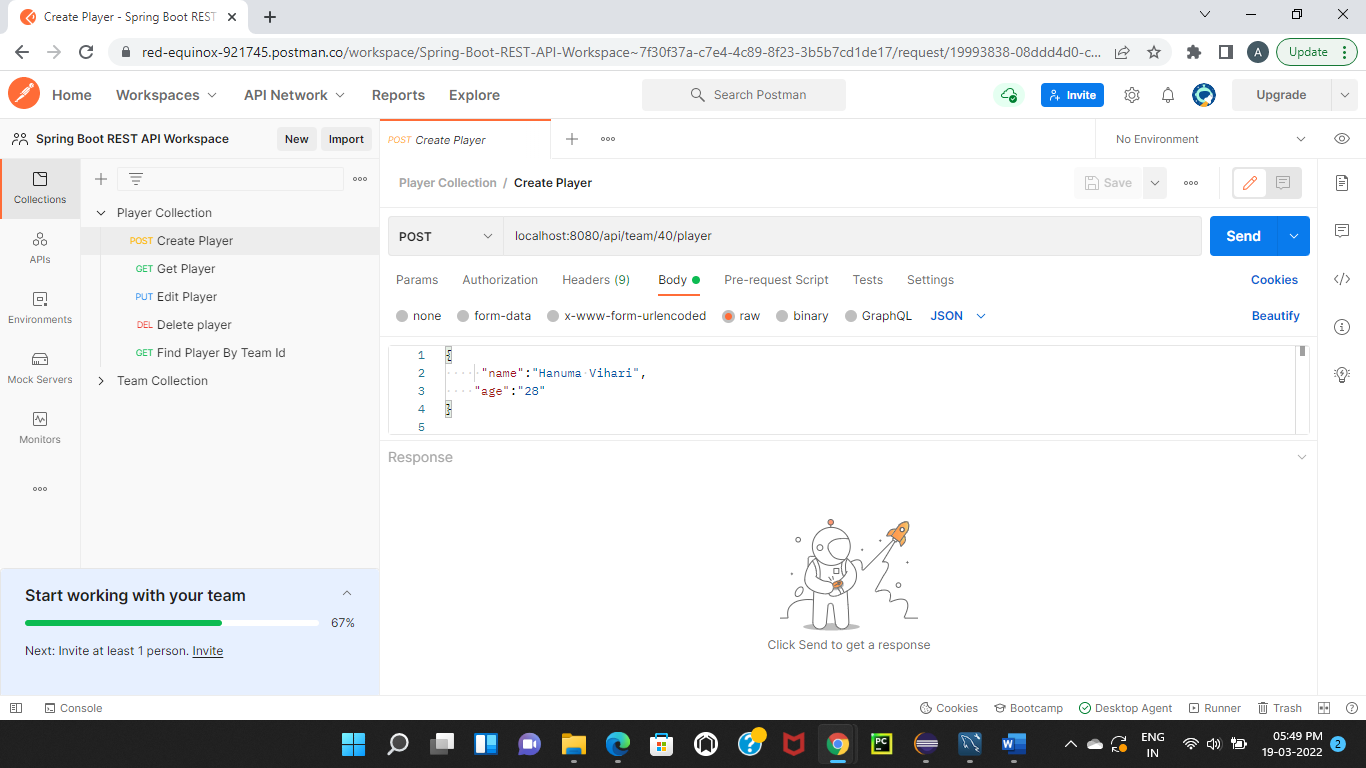
The created team data has been saved in database as shown in below.



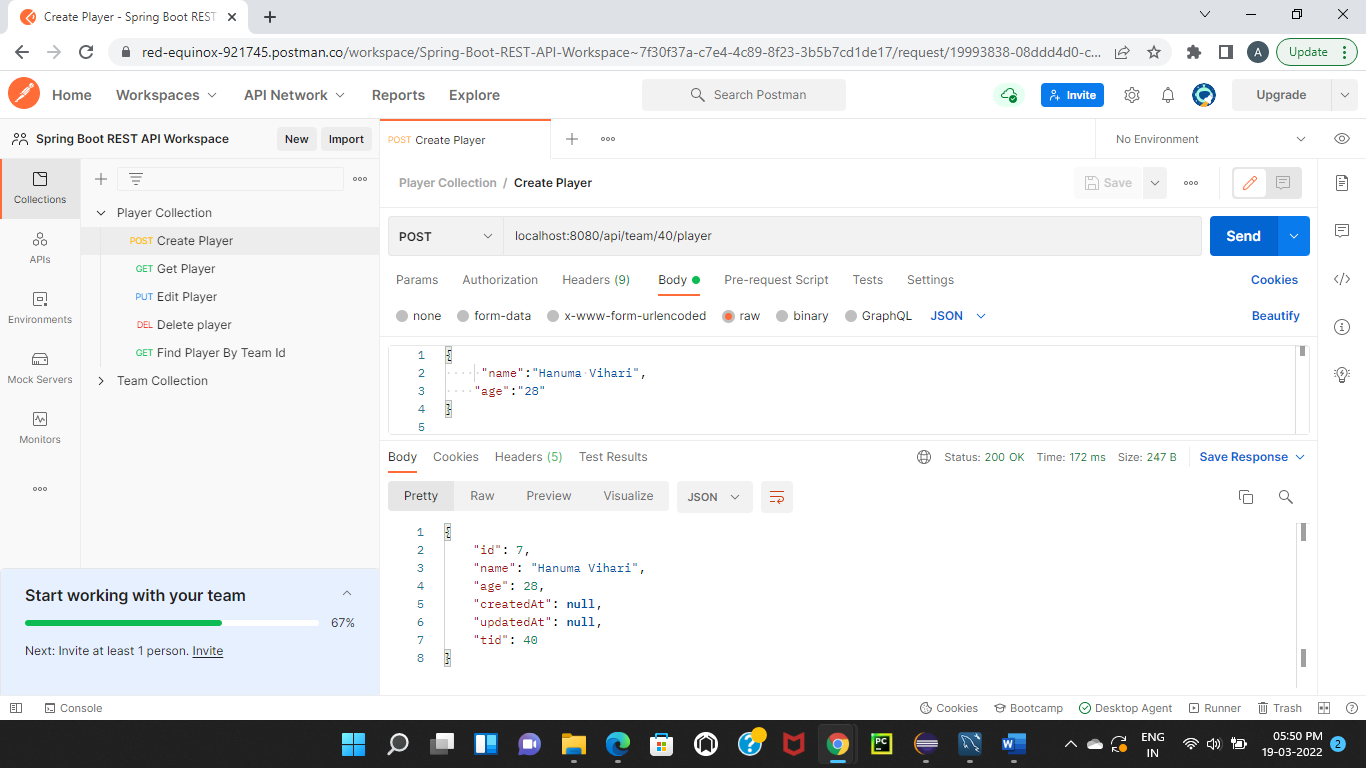
**Player**

Similarly, to create a Player, I have sent POST request for the URL <https://localhost:8080/api/team/40/player>. In the URL, 40 is team id under which player has to create.

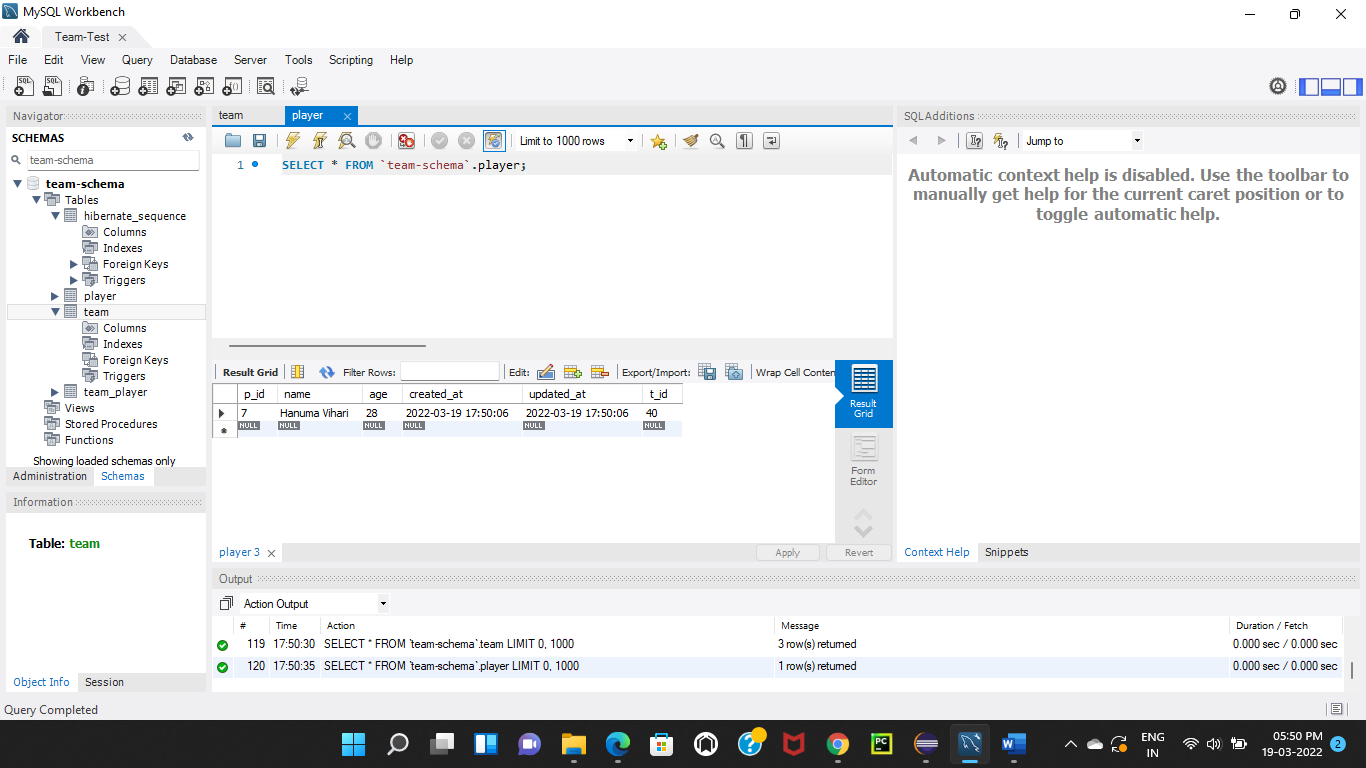
Below is the image of before sending POST request to create player.



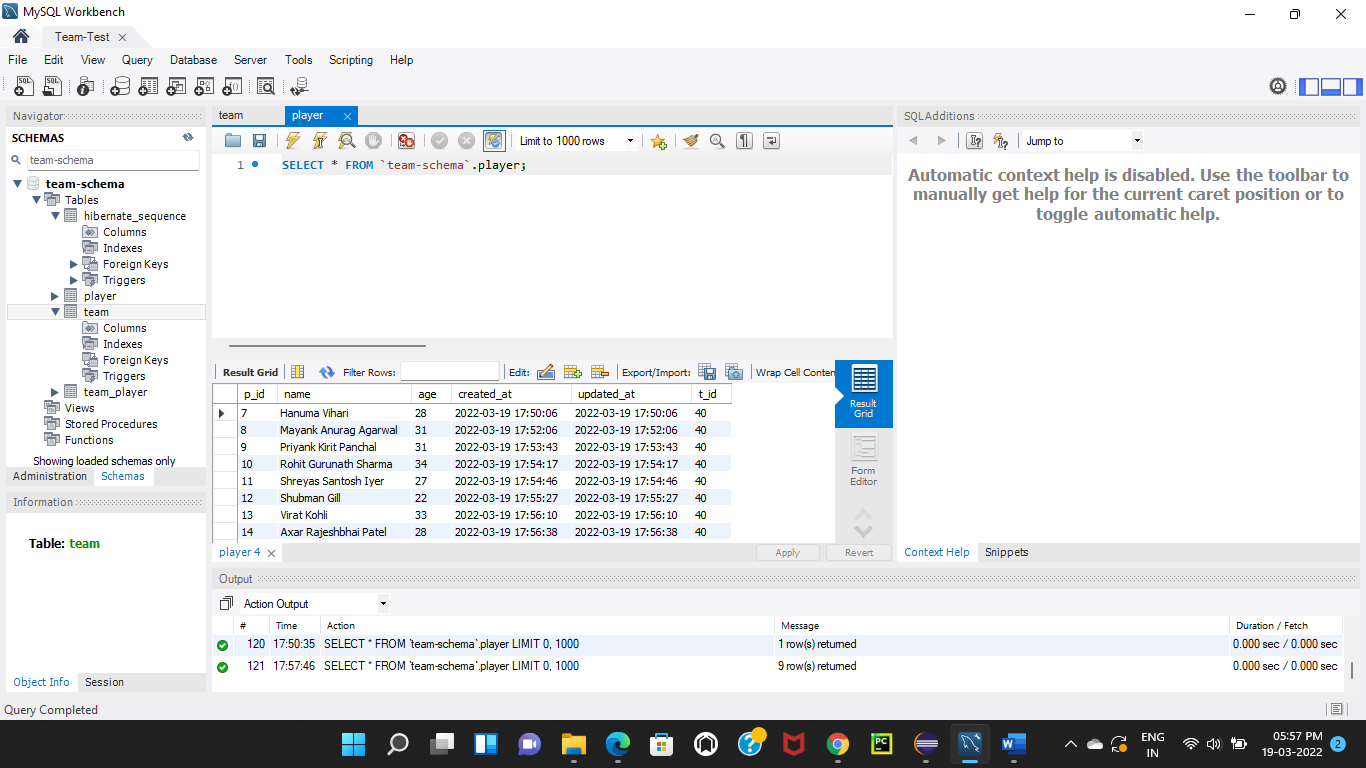
Below is the image of after sending POST request and getting response of created player



The created player data has been saved in database as shown in below.



Multiplt player has been saved in the player table which is mapped with team id.

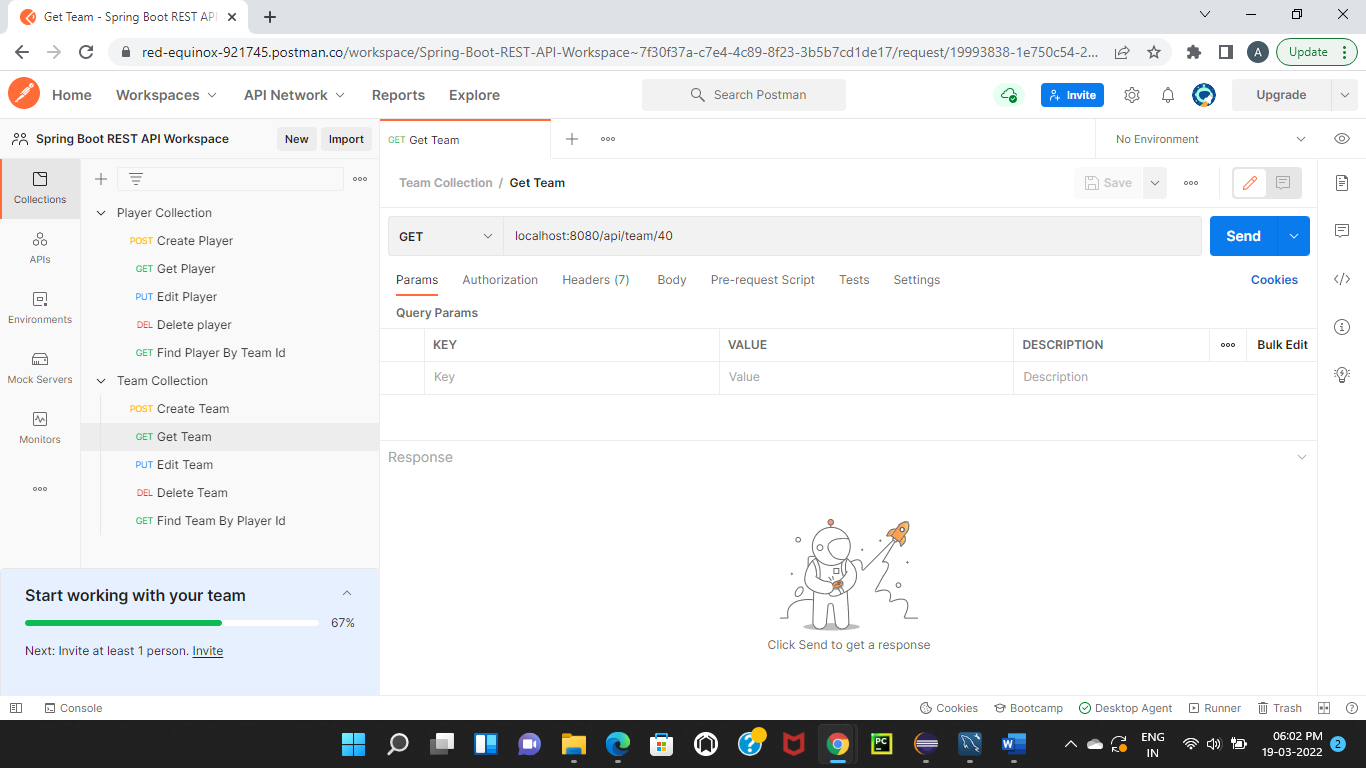


**Find By Id**

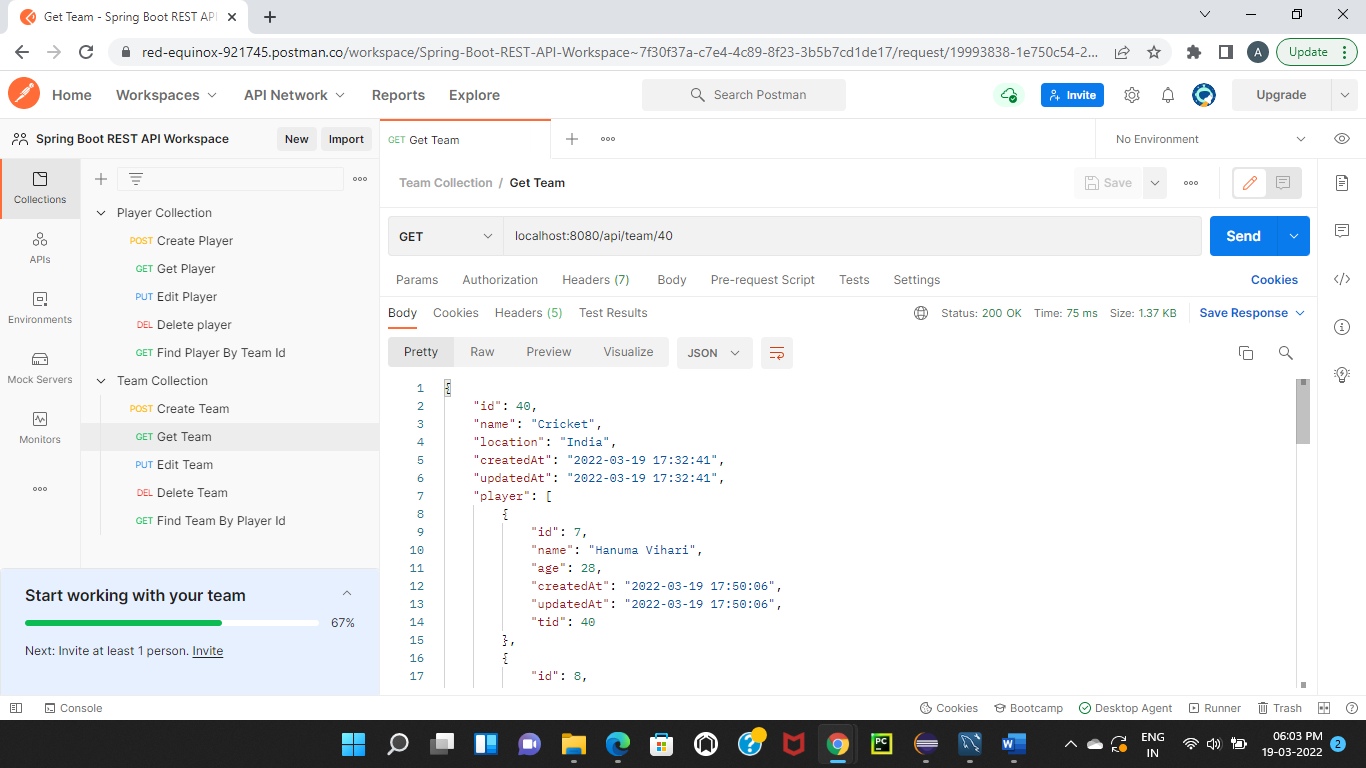
**Team**

To find a Team by its id, I have sent GET request for the URL <https://localhost:8080/api/team/40> and left body empty. In the URL, I have given RequestMapping as “/api/team/40” which is working for read endpoint where 40 is id of a team to be find.

Below is the image of before sending GET request to find a team.



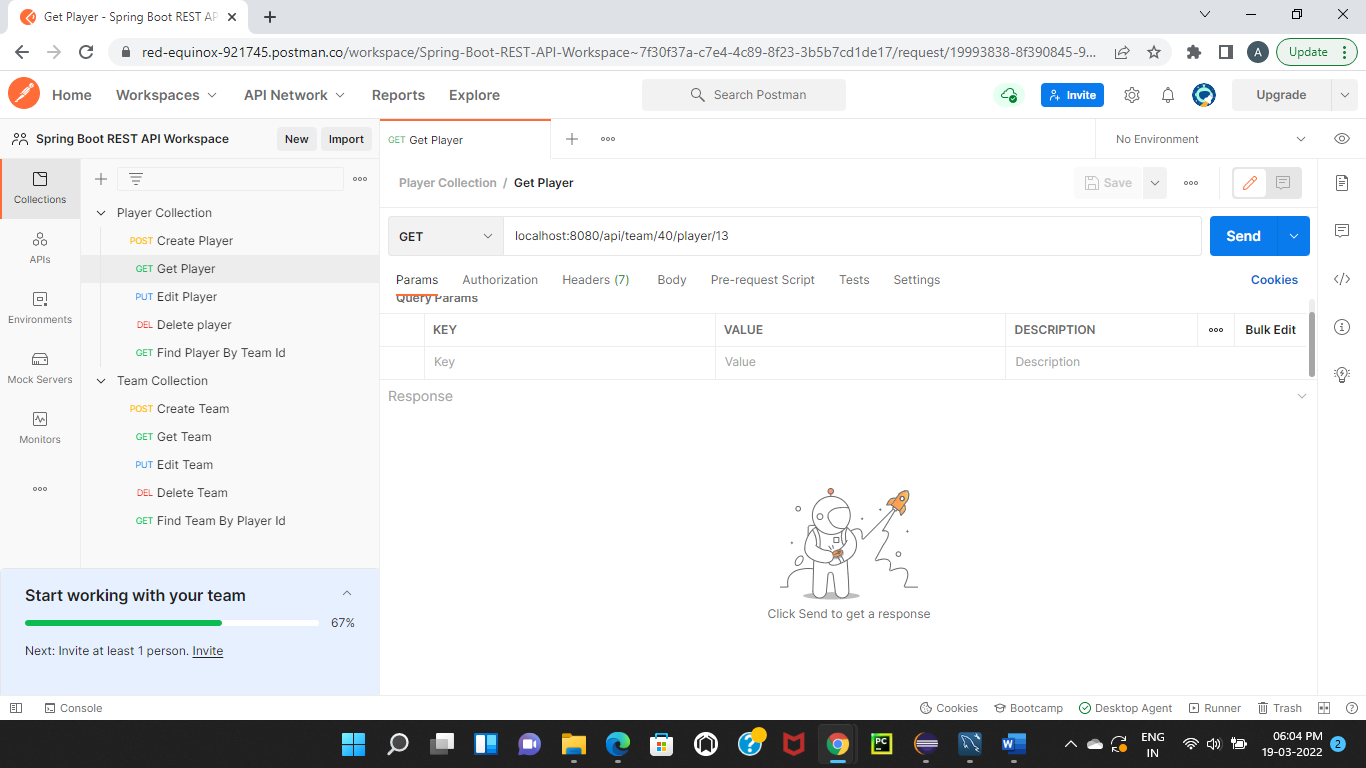
Below is the image of after sending GET request and getting response that a team has been found.



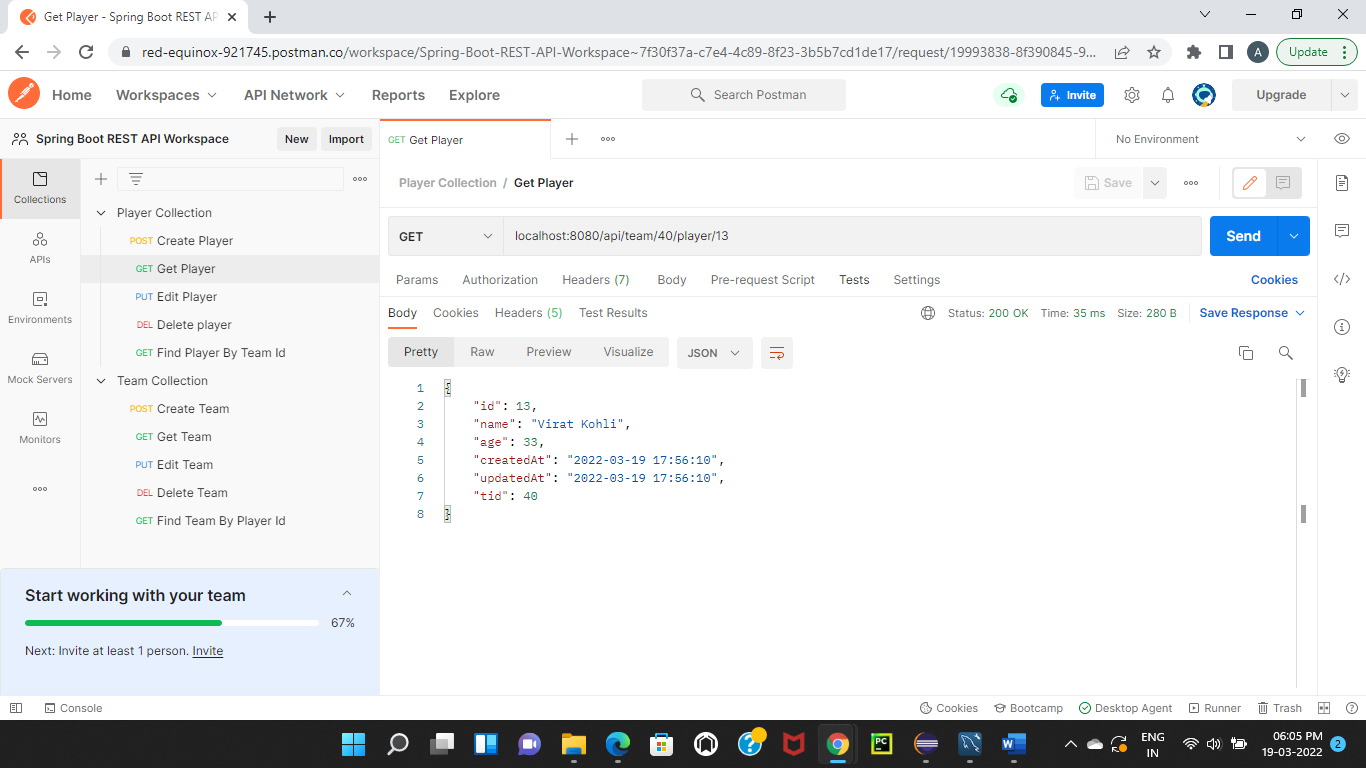
**Player**

To find a Player by its id, I have sent Get request for the URL <https://localhost:8080/api/team/40/player/30> and left the body empty. In the URL, I have given RequestMapping as “/api/team/40/player/30” which is working for read endpoint where 40 is id of a team and 30 is id of a player to be find.

Below is the image of before sending GET request to find a player.



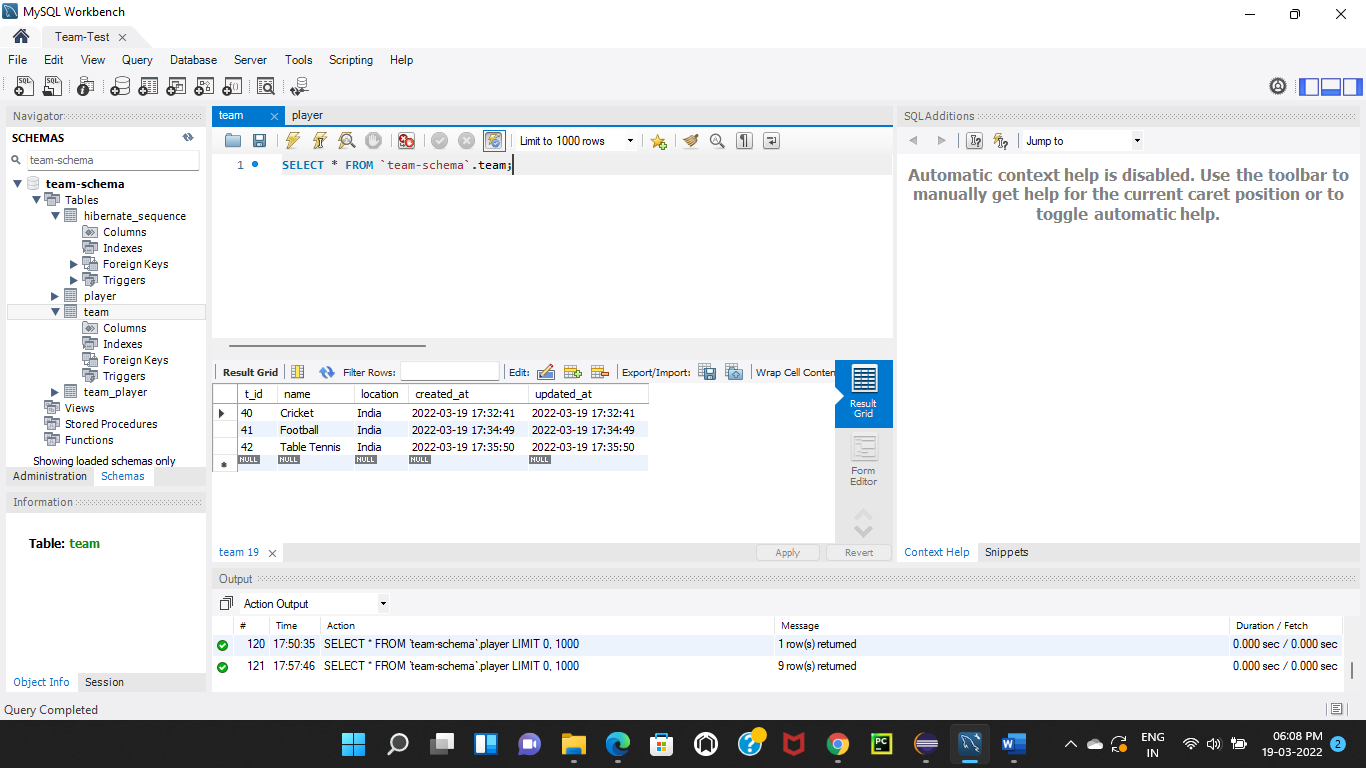
Below is the image of after sending GET request and getting response that a team has been found.



**Update**

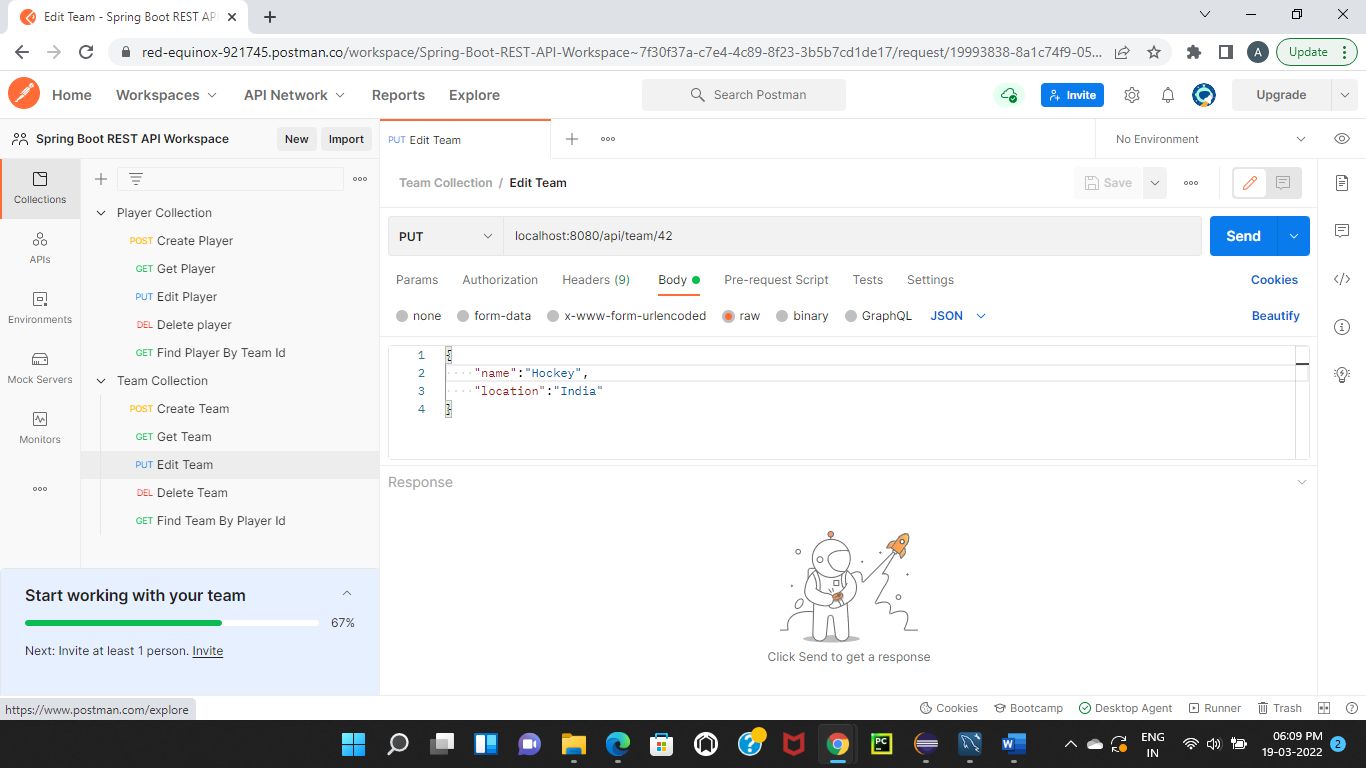
**Team**

Before attempt to update the data, below is the image of team database.

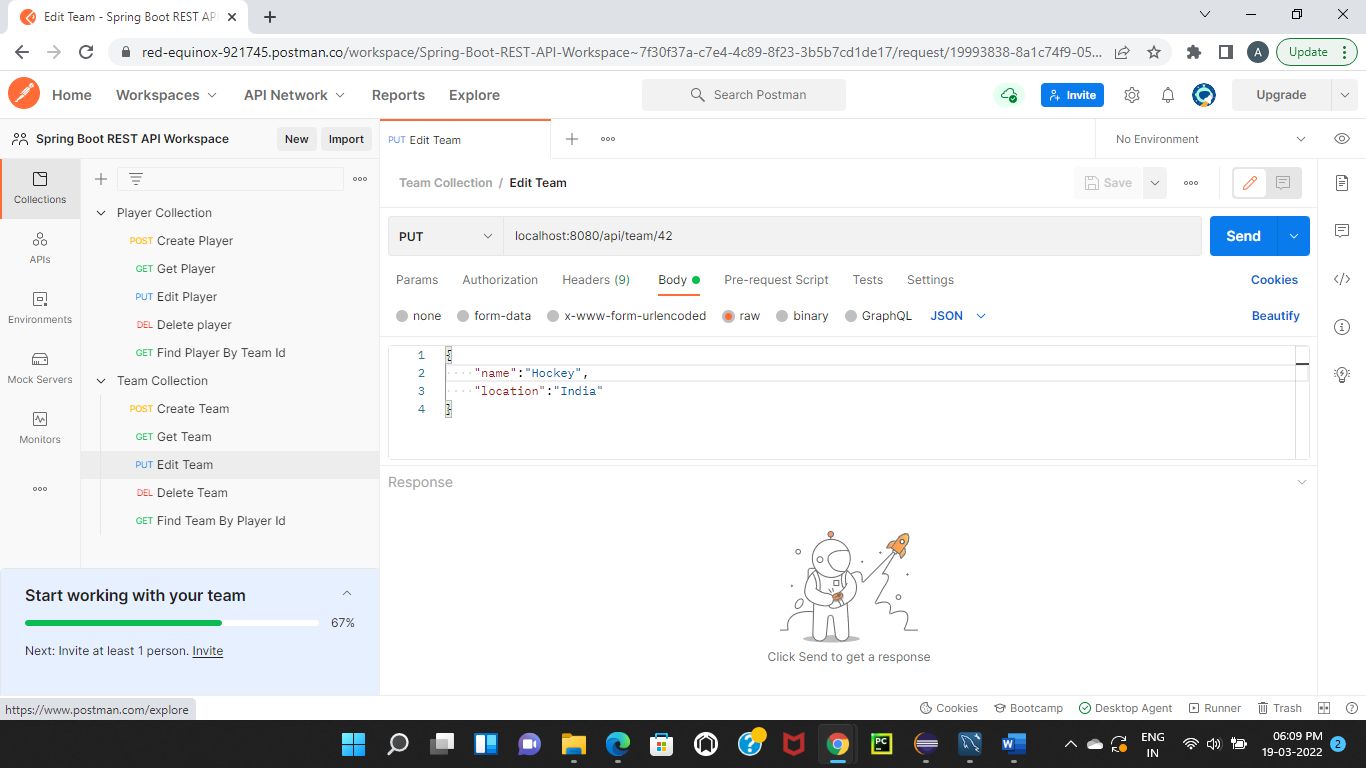


To update team, I have sent PUT request for the URL <https://localhost:8080/api/team/42>. In the URL, I have given RequestMapping as “/api/team/42” which is working for update endpoint where 42 is id of a team to be updated.

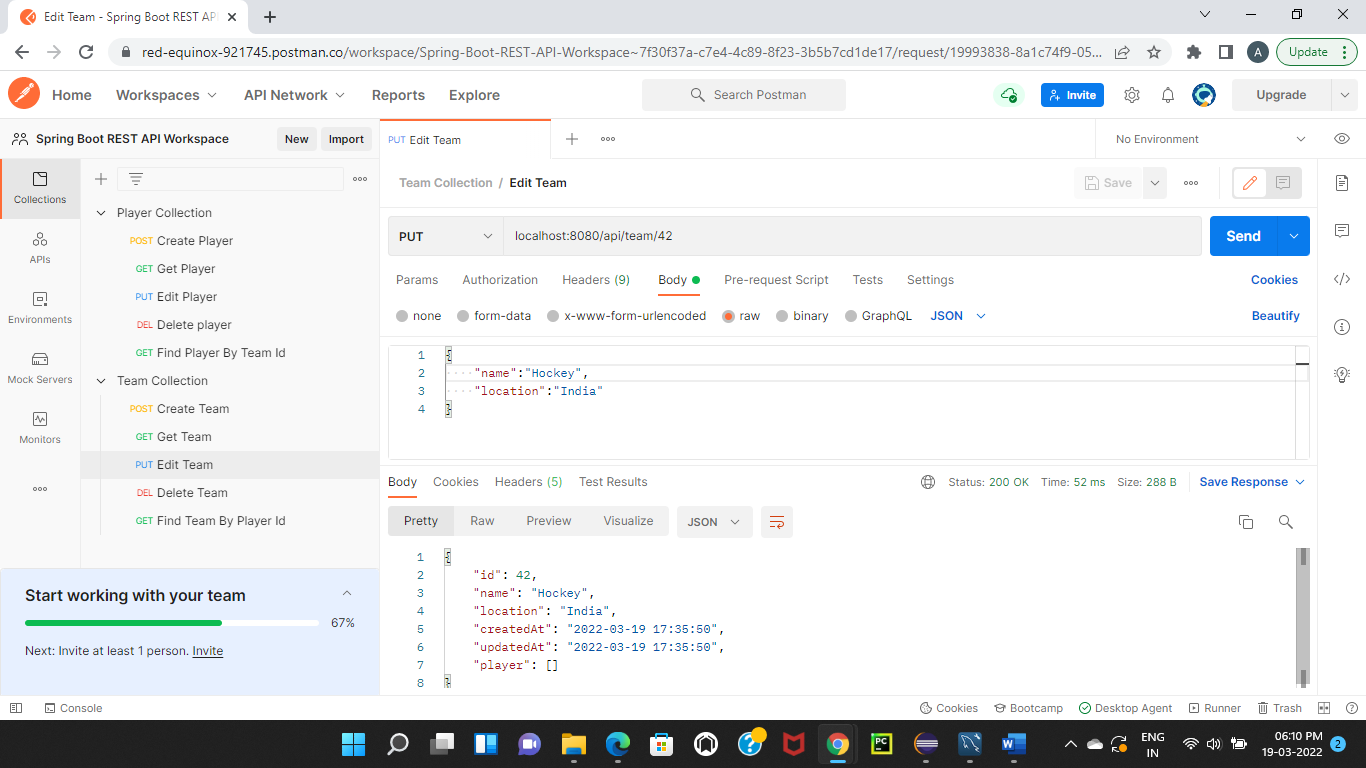
Entered below data to send request body to the given URL endpoint. These are the Team data that will be updated into the database.



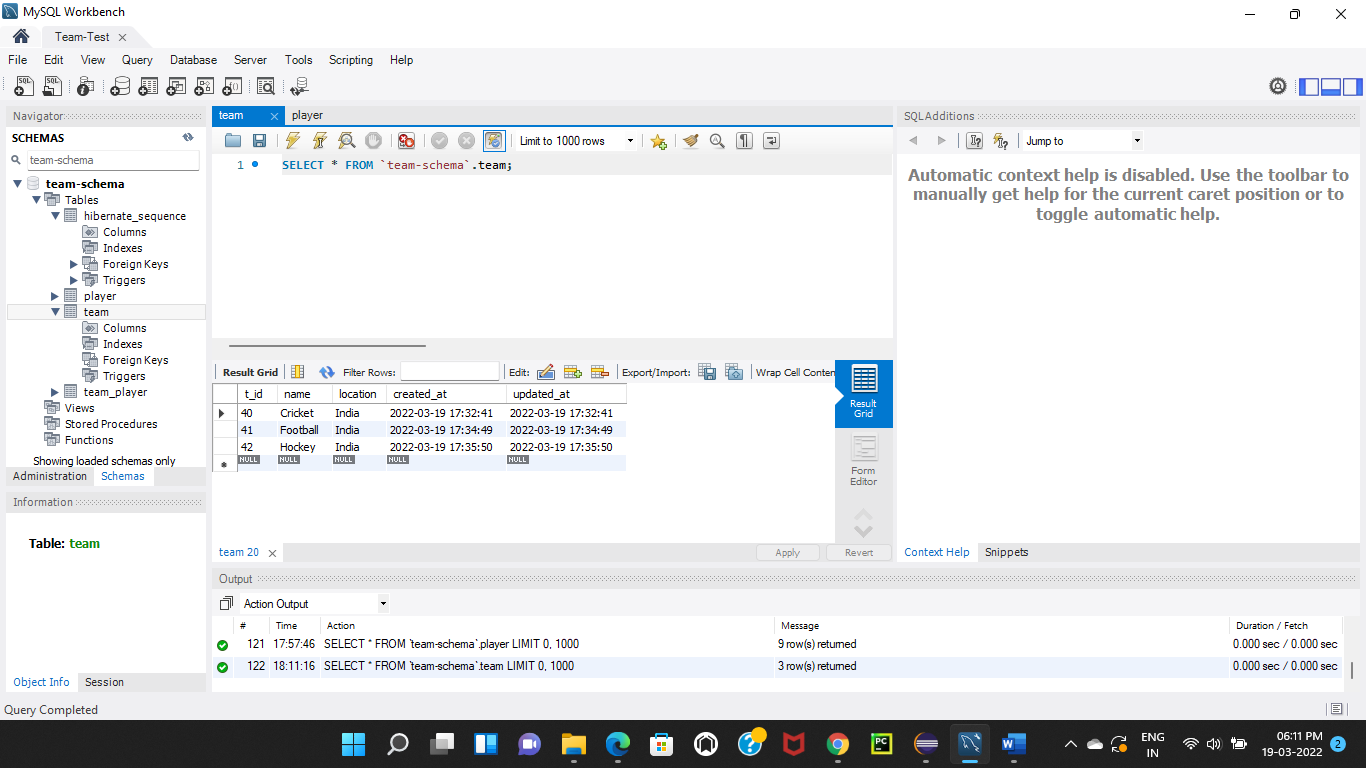
Below is the image of before sending PUT request to update a team.



Below is the image of after sending PUT request and getting response as updated team details.

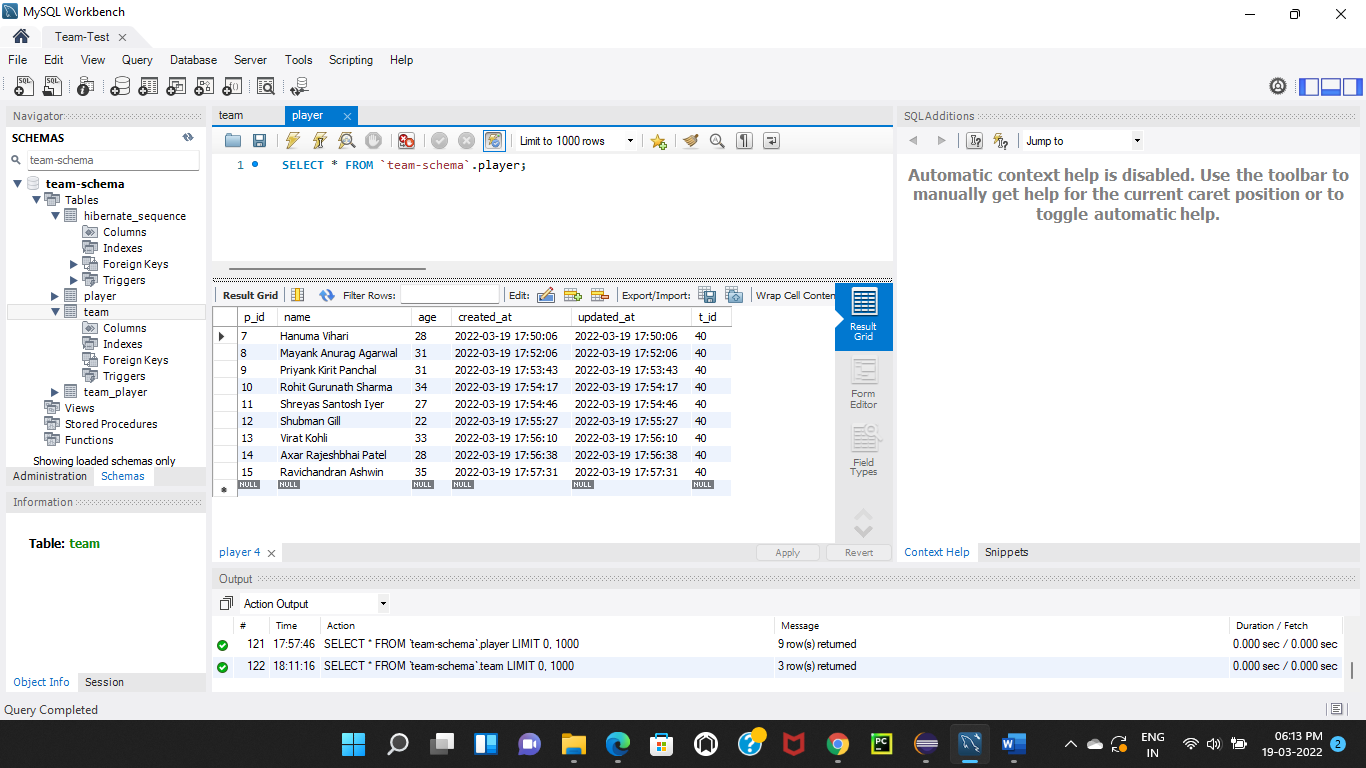


Below is the image of updated team database where id is 42.



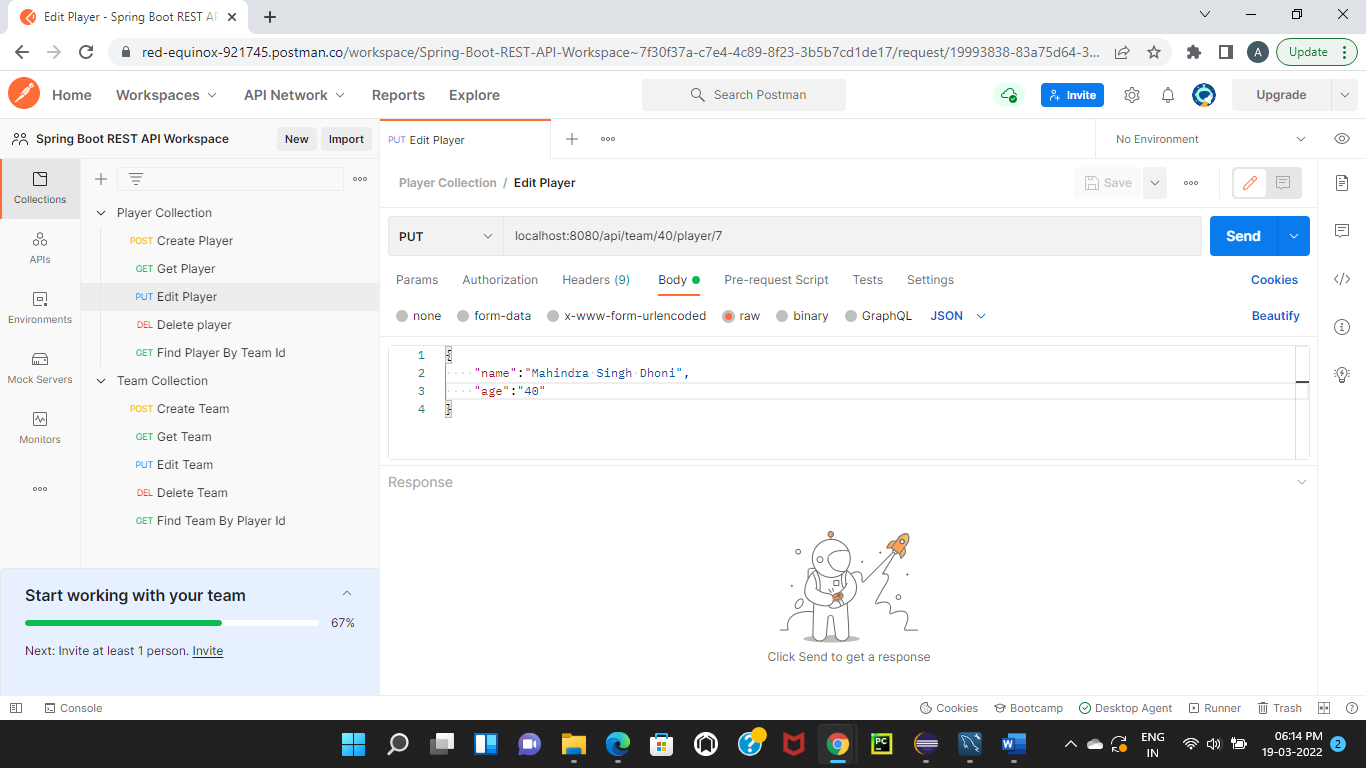
**Player**

Before attempt to update the data, below is the image of player database.

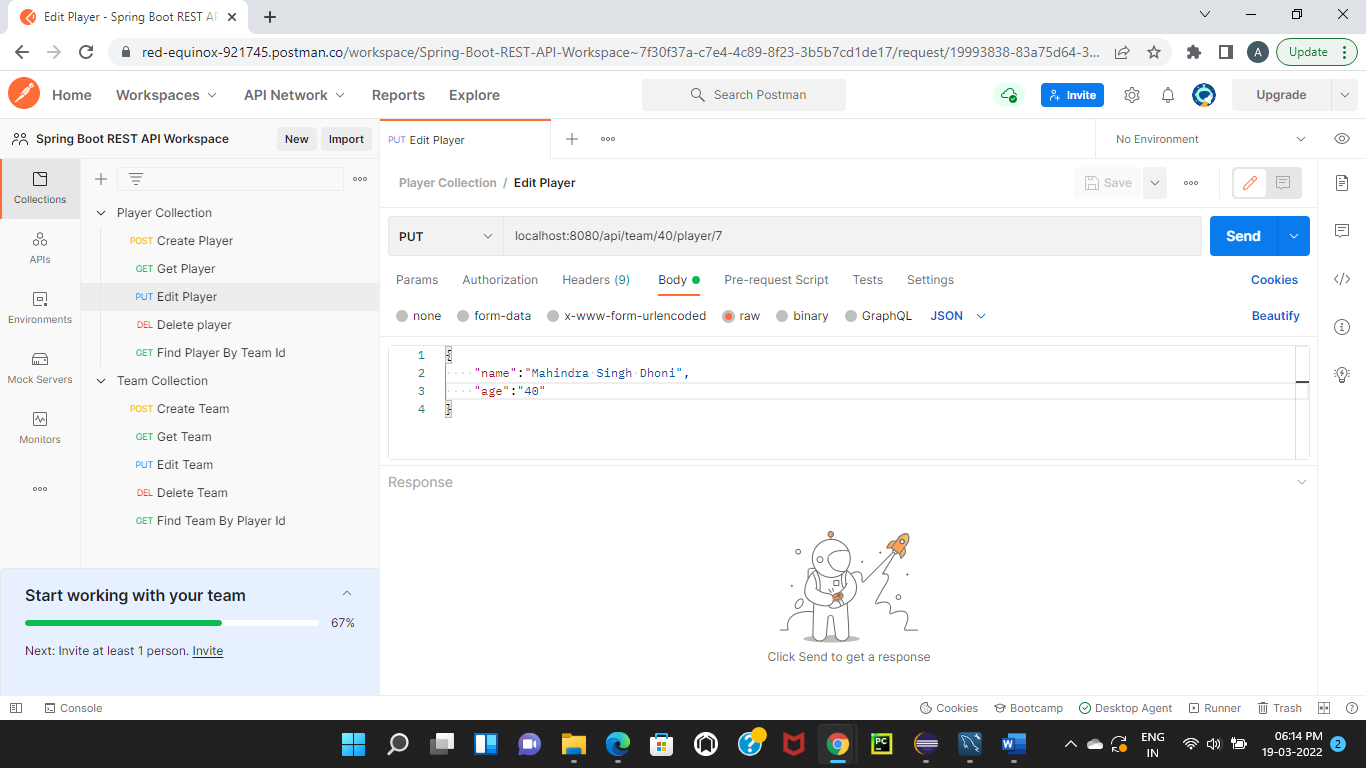


To find a Player by its id, I have sent PUT request for the URL <https://localhost:8080/api/team/40/player/7>. In the URL, I have given RequestMapping as “/api/team/40/player/7” which is working for update endpoint where 40 is id of a team and 7 is id of a player to be updated.

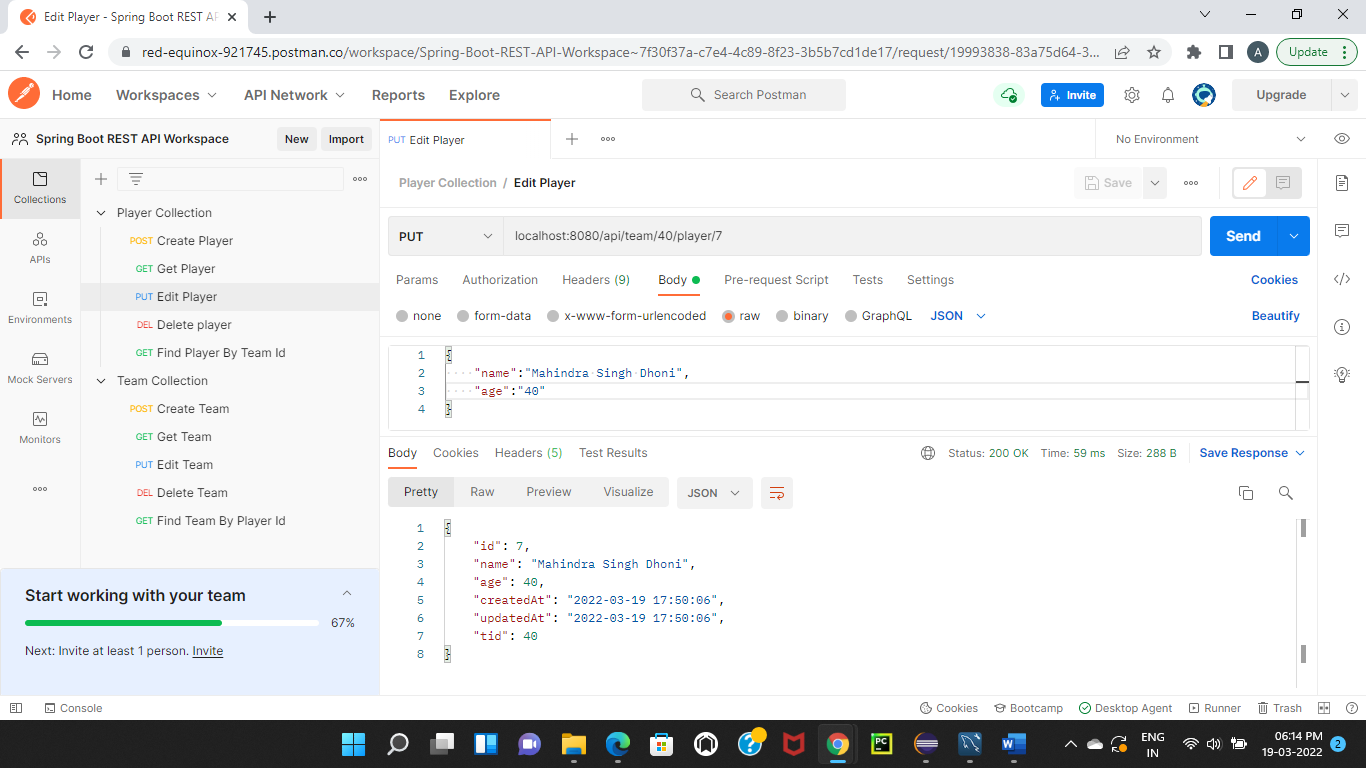
Entered below data to send request body to the given URL endpoint. These are the Player data that will be updated into the database.



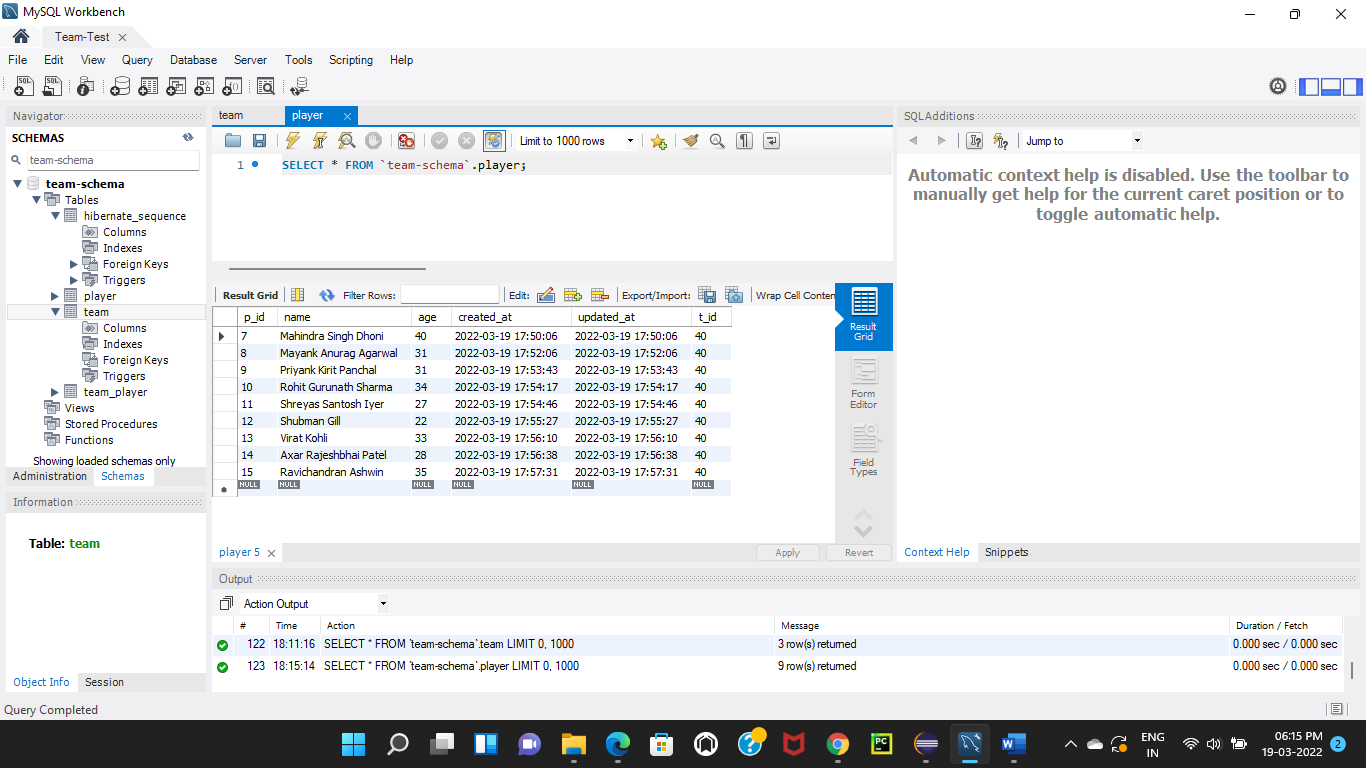
Below is the image of before sending PUT request to update a player.



Below is the image of after sending PUT request and getting response as updated player details.



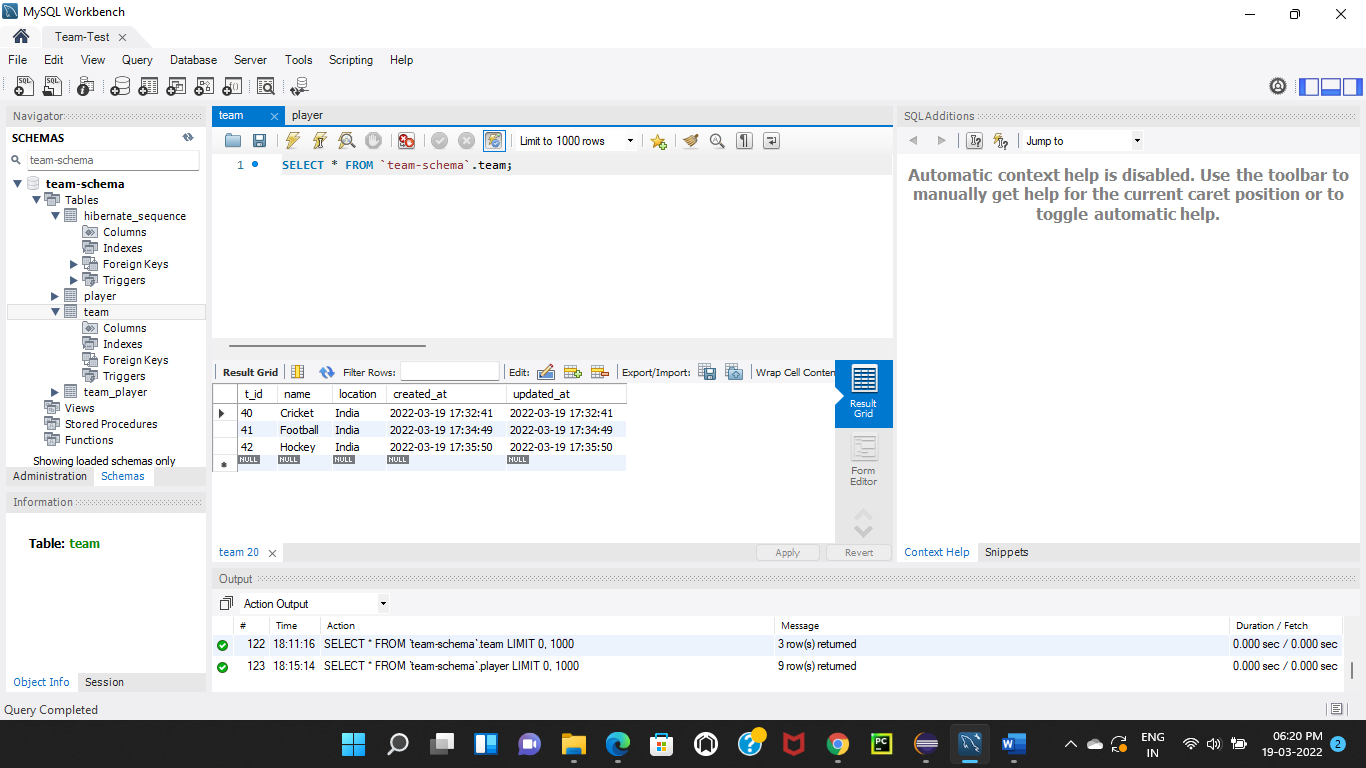
Below is the image of updated player database where player id is 7.

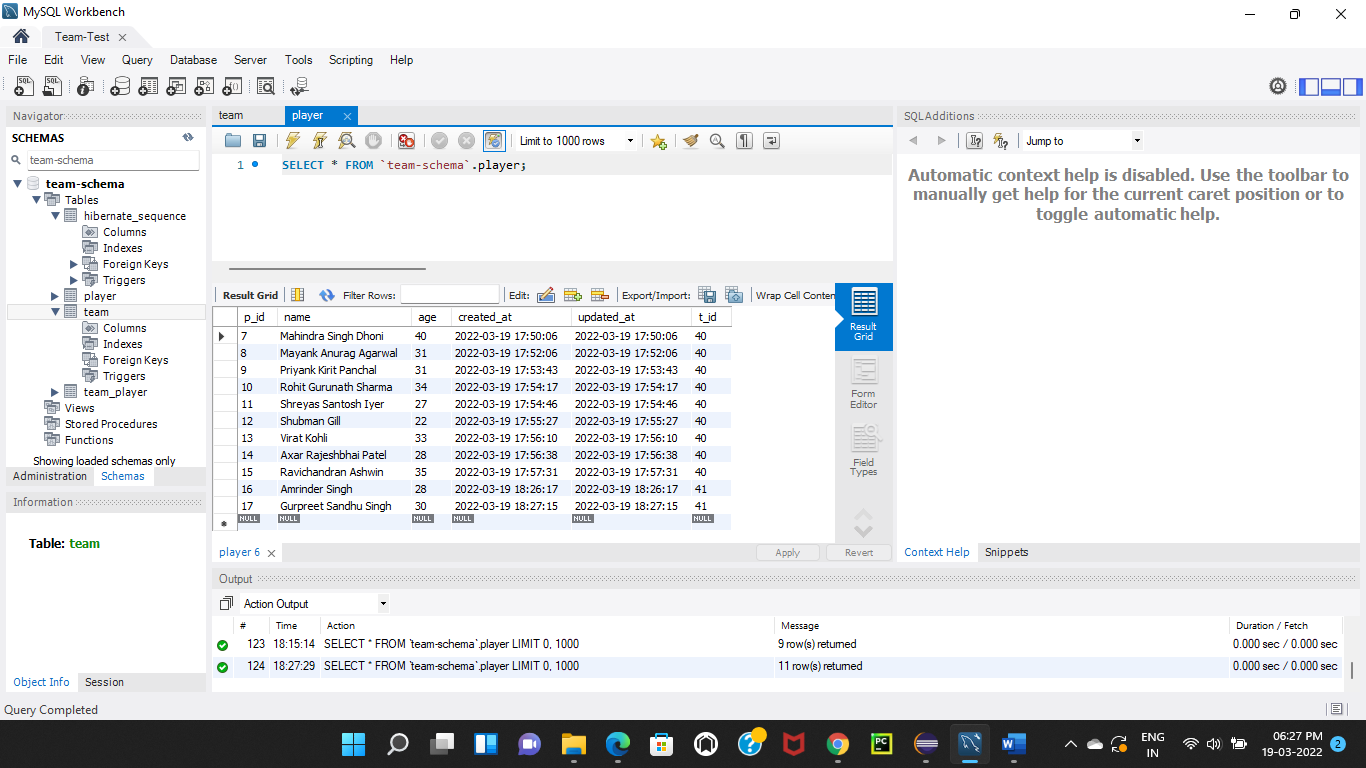


**Delete**

**Team**

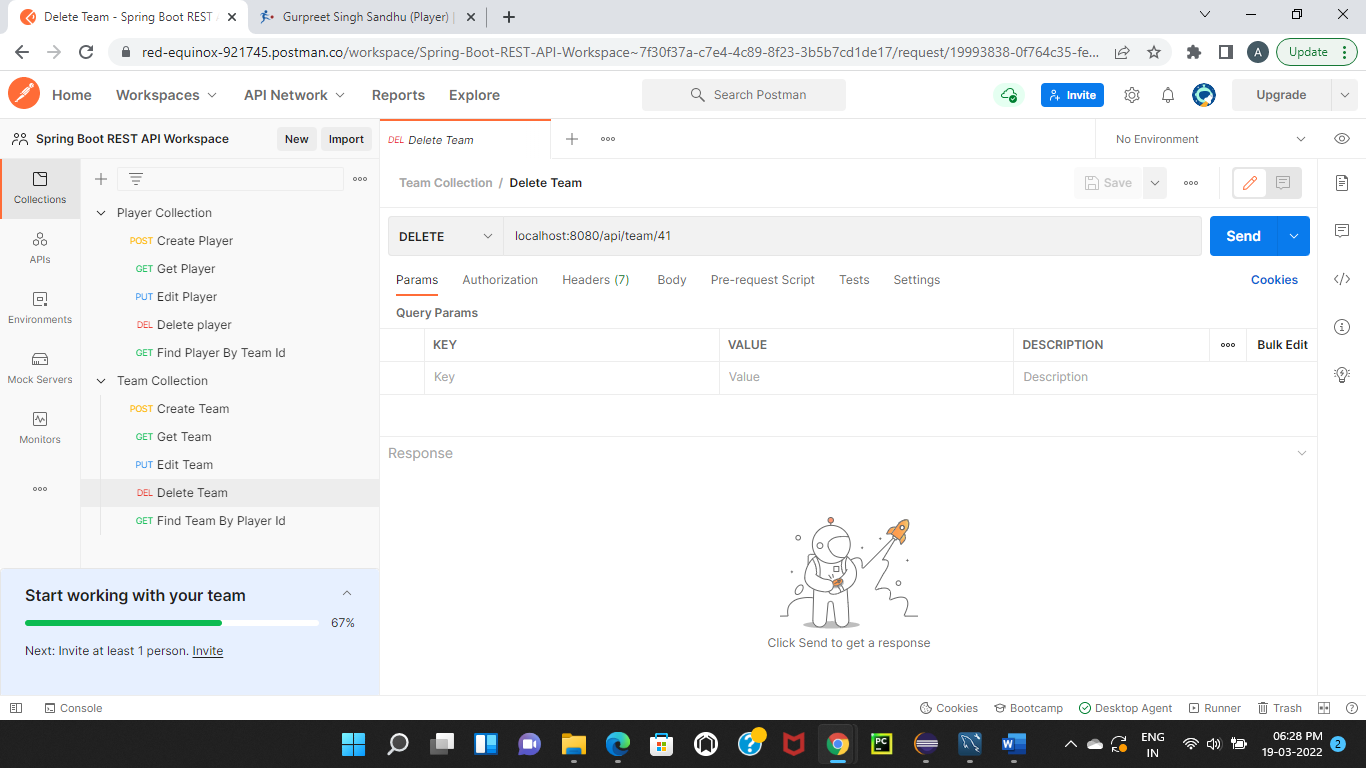
Before attempt to delete the data, below is the image of team as well as player database. Player data will also be deleted if team data is deleted.





To delete a team, I have sent DELETE request for the URL <https://localhost:8080/api/team/41> and left the body empty. In the URL, I have given RequestMapping as “/api/team/41” which is working for delete endpoint where 41 is id of a team to be deleted.

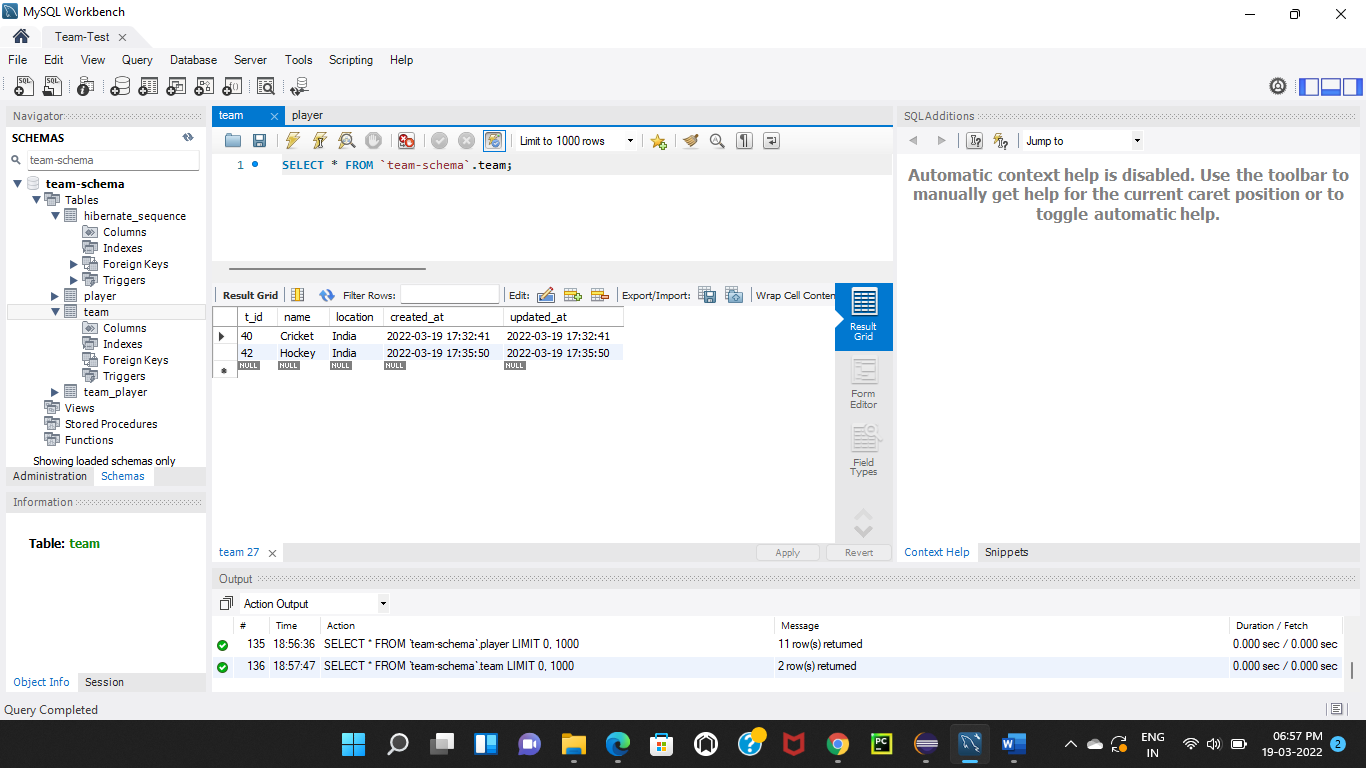
Below is the image of before sending DELETE request to delete a team.

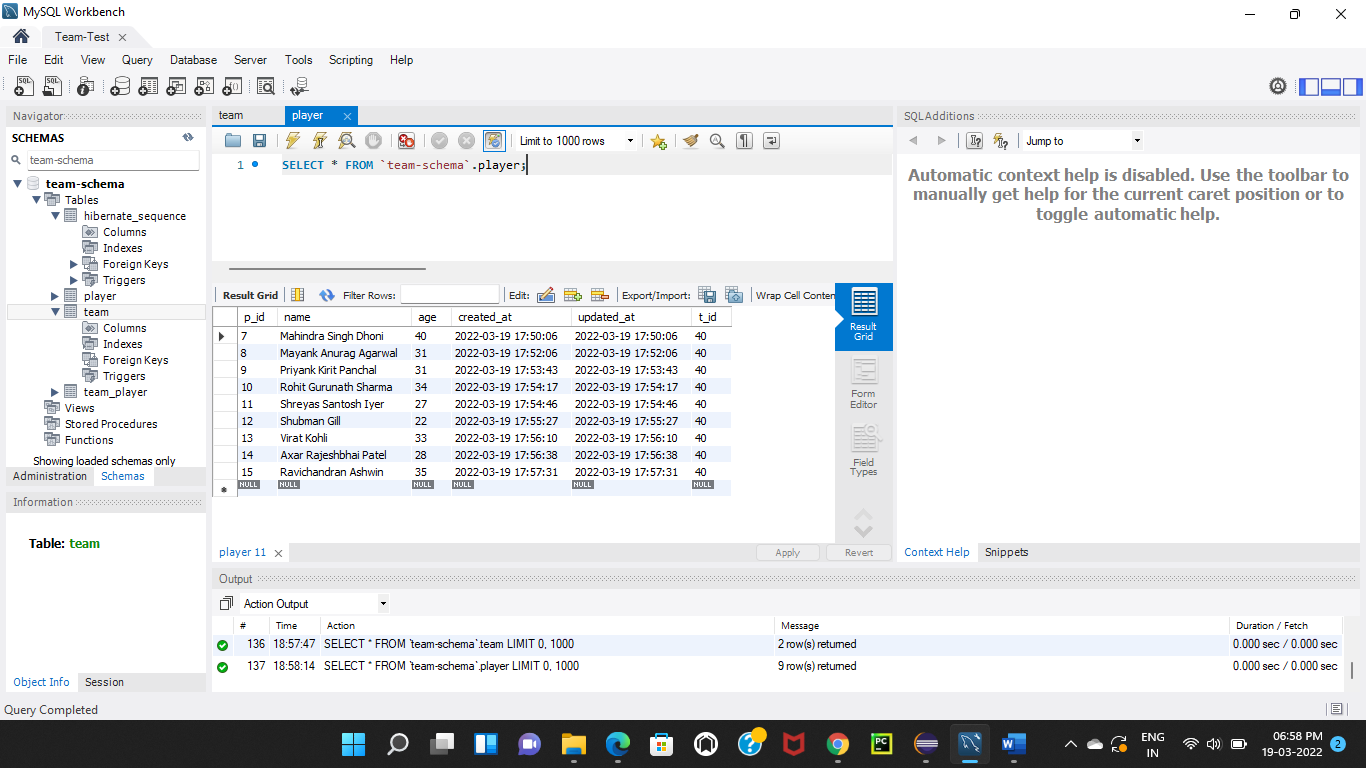


Below is the image of after sending DELETE request and getting HTTP status as 200 Ok.



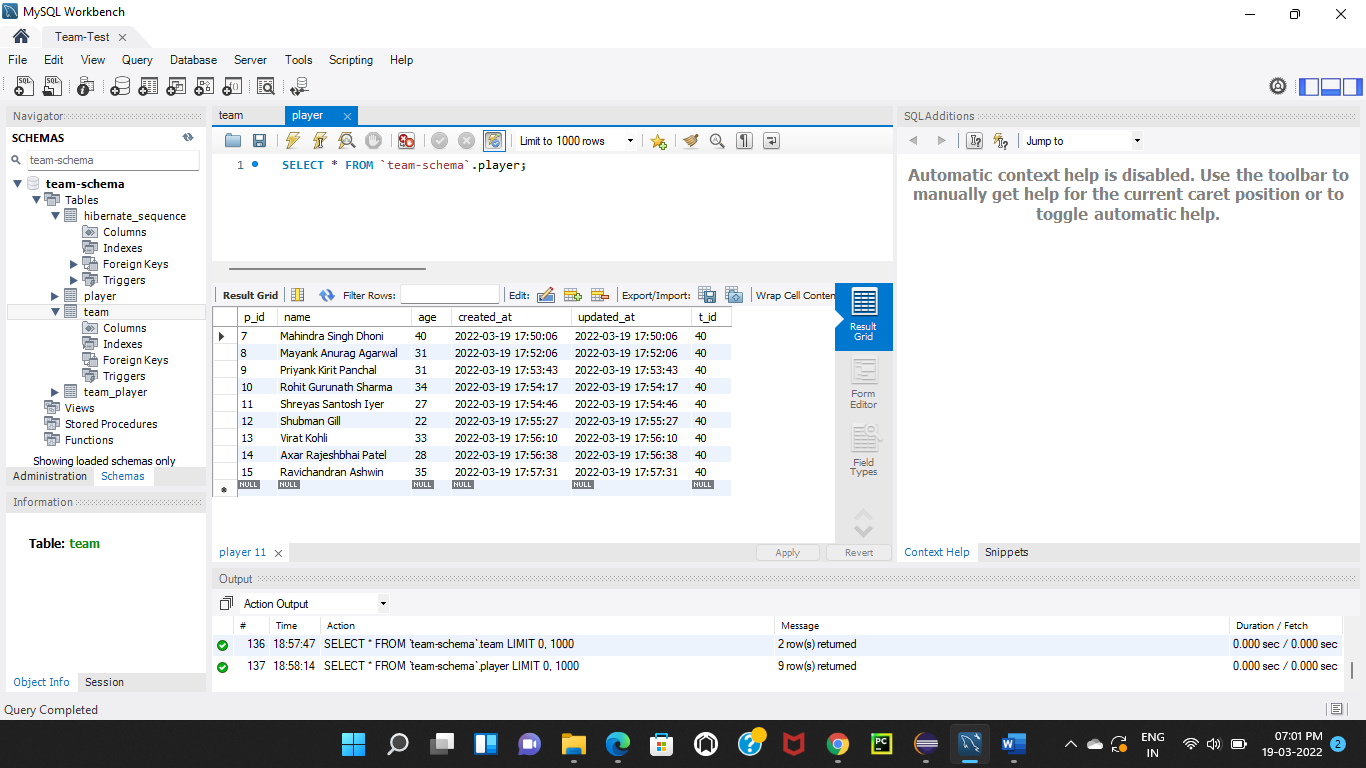
Here is the database of team and player where team id 41 has been deleted from team database and player data has also been deleted as team id is mapped with player database.





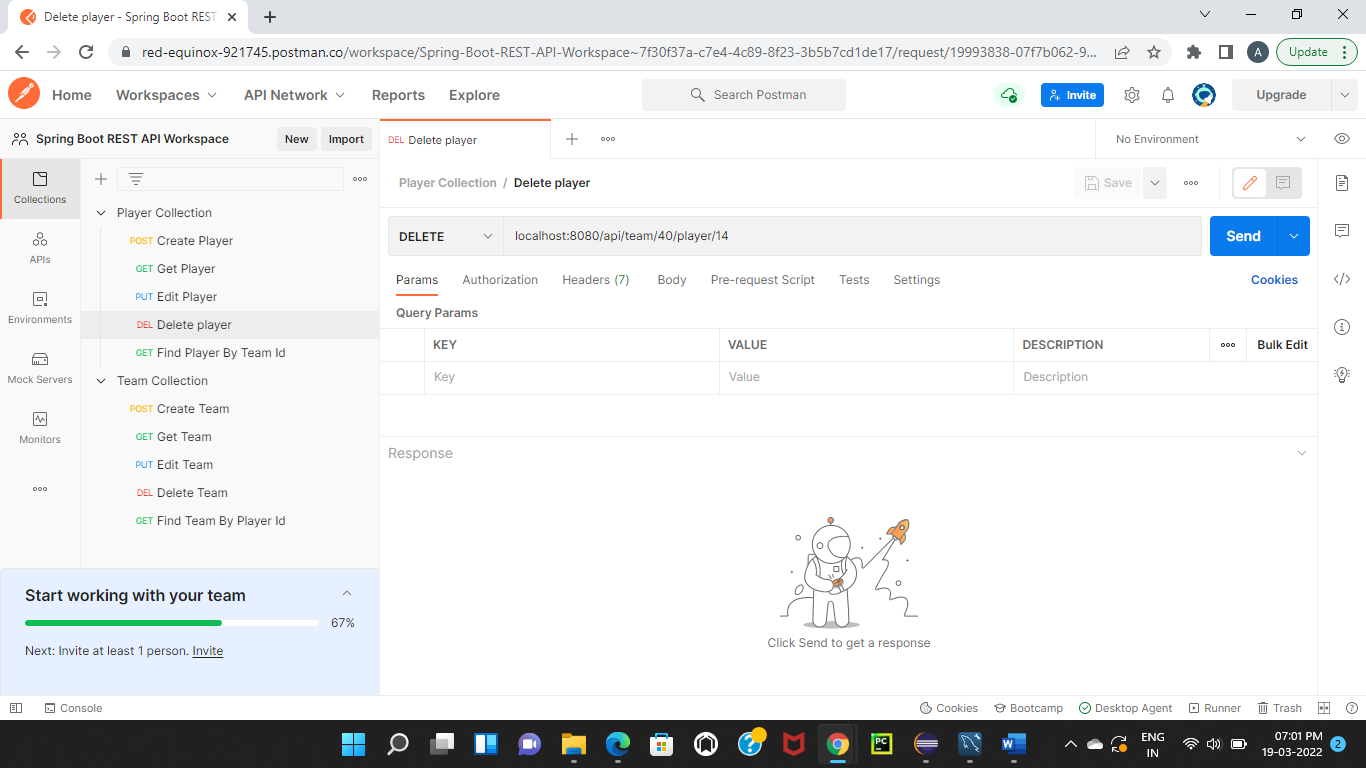
**Player**

Before attempt to delete the data, below is the image of player database.

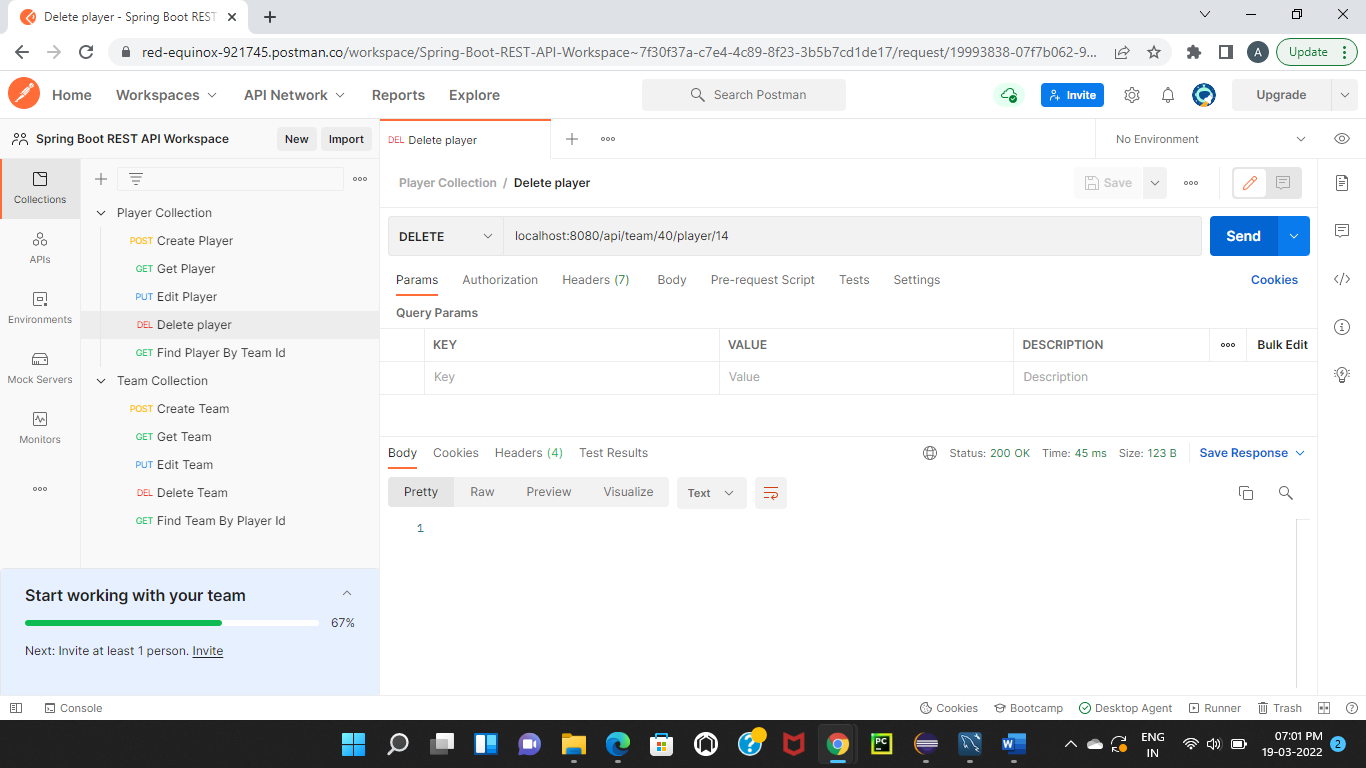


To delete a team, I have sent DELETE request for the URL <https://localhost:8080/api/team/40/player/14> and left the body empty. In the URL, I have given RequestMapping as “/api/team/40/player/30” which is working for delete endpoint where 40 is id of a team and 14 is id of a player to be delete.

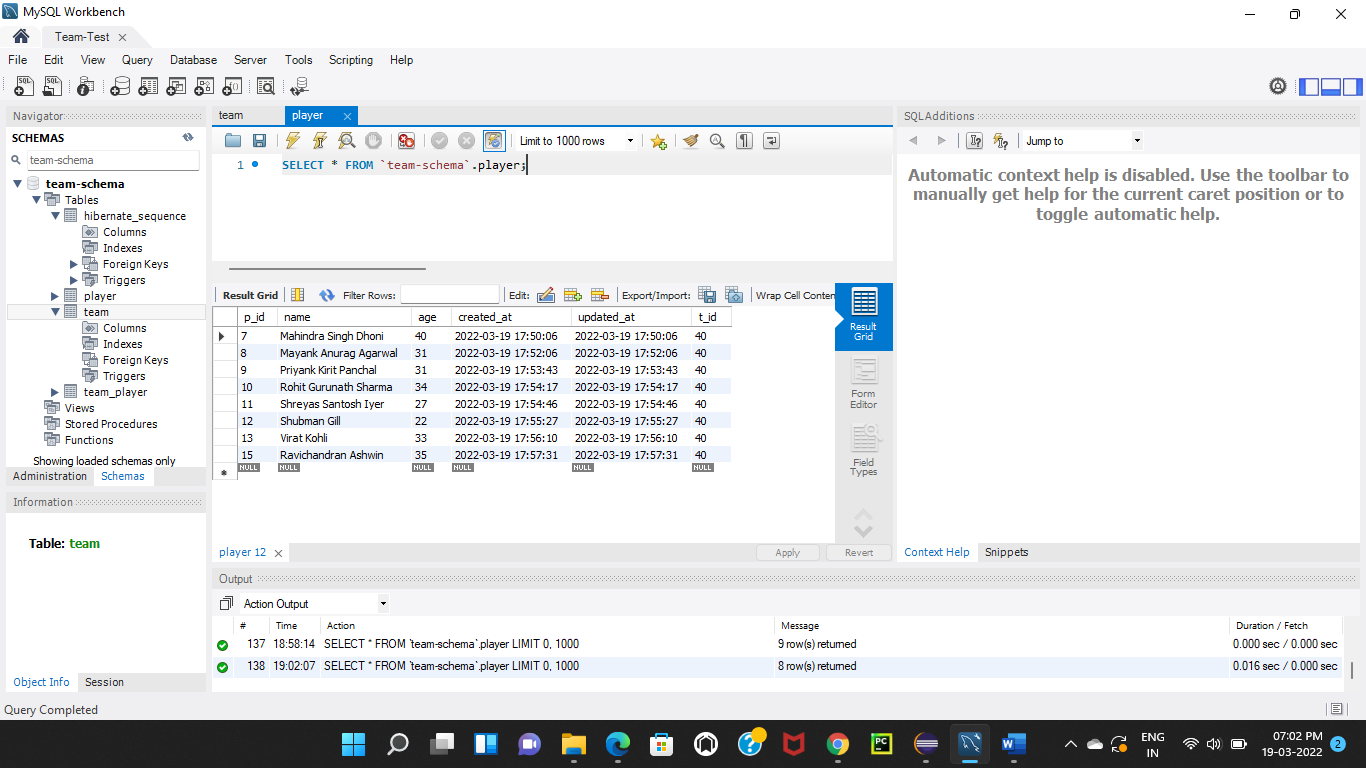
Below is the image of before sending DELETE request to delete a player.



Below is the image of after sending DELETE request and getting HTTP status as 200 Ok.



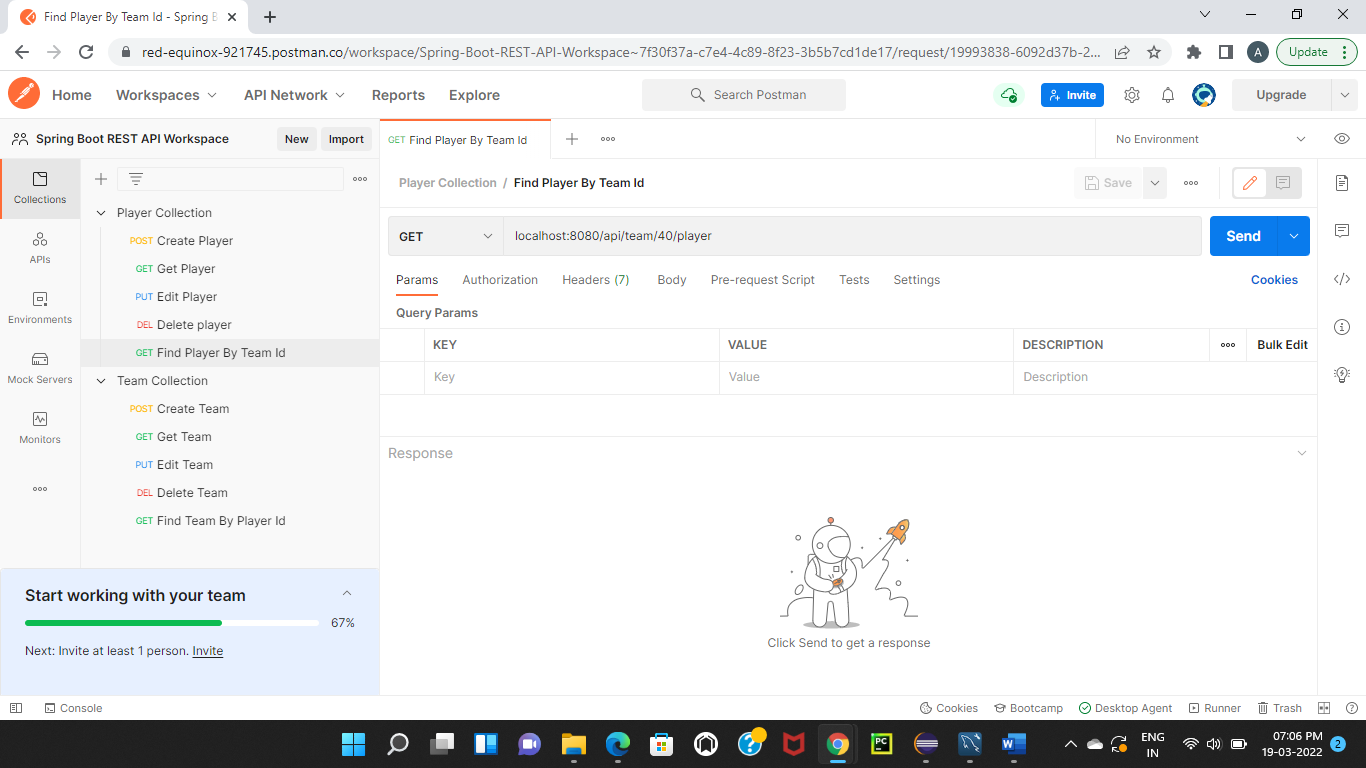
Here is the database of player where player id 14 has been deleted from player database.



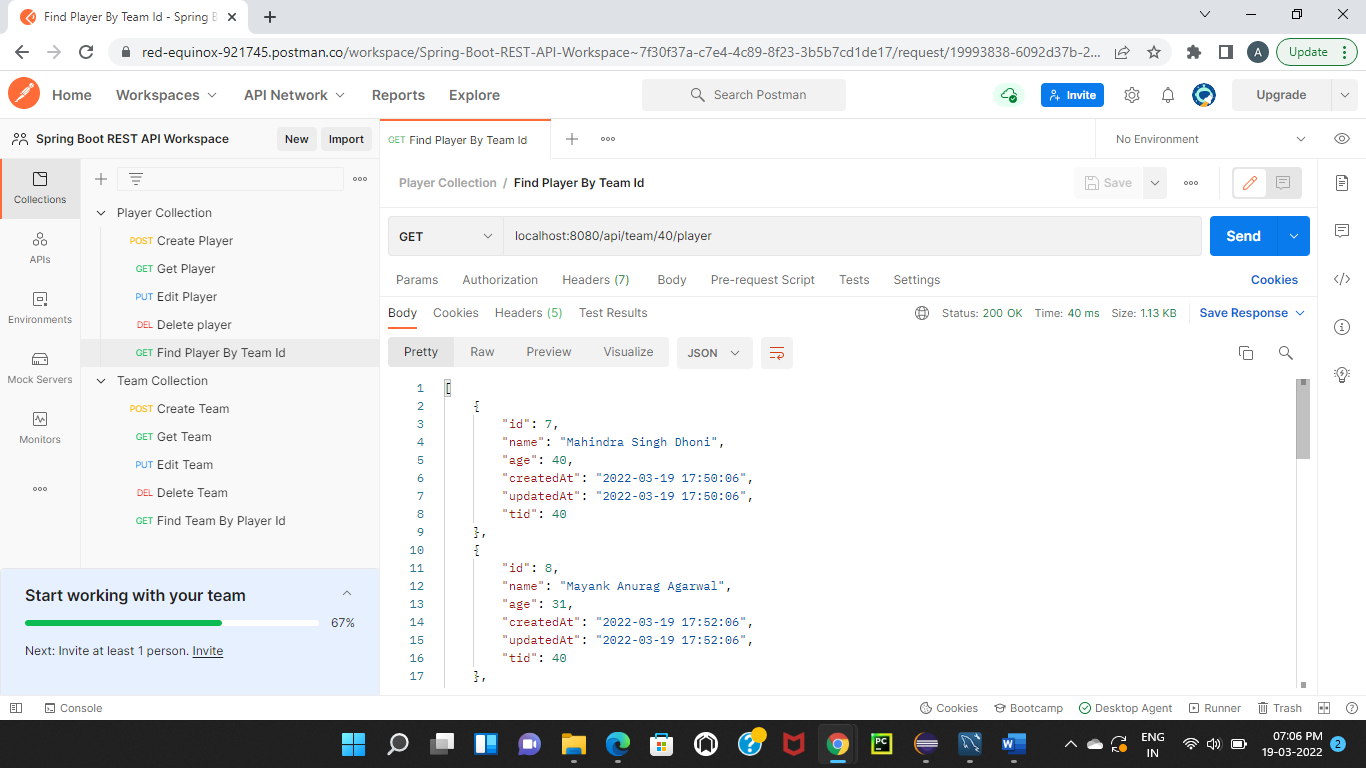
**Find players by team id**

To find a Player by team id, I have sent GET request for the URL <https://localhost:8080/api/team/40/player> and left body empty. In the URL, I have given RequestMapping as “/api/team/40/player” which is working for read endpoint where 40 is id of a team using to find player.

Below is the image of before sending GET request to find a player.



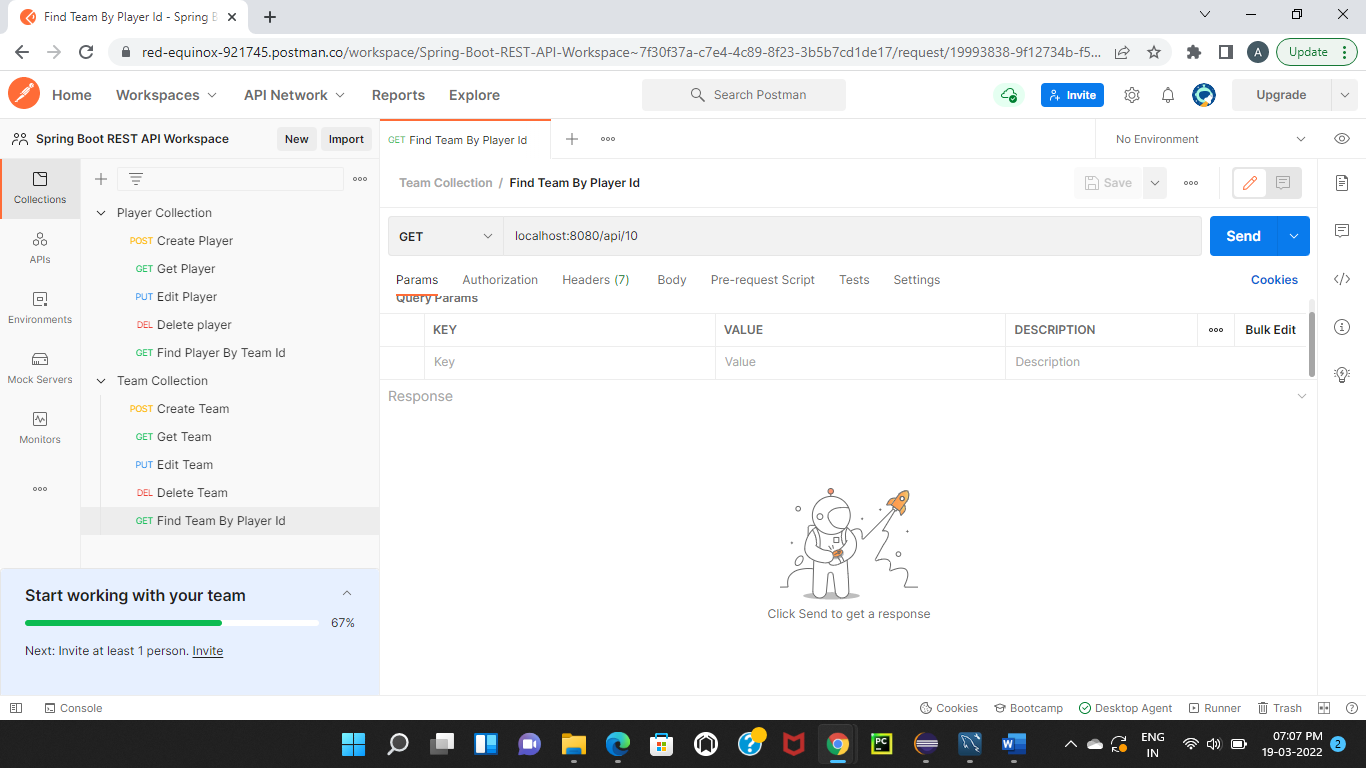
Below is the image of after sending GET request and getting response that a player has been found.



**Find team by player id**

To find a Team by player id, I have sent GET request for the URL <https://localhost:8080/api/10> and left body empty. In the URL, I have given RequestMapping as “/api/team/10” which is working for read endpoint where 10 is id of a player using to find team.

Below is the image of before sending GET request to find a team.



Below is the image of after sending GET request and getting response that a team has been found.

