HW12 – MYSQL

goteam database

The GoTeam application is a sophisticated application that leverages Spring Boot for its backend services, mySQL as its database, and is linked to a frontend component ensuring real-time data synchronization. Below is a detailed walkthrough of the project, including database modifications, backend configuration, and how Maven is utilized to manage the project’s lifecycle.

**Steps done to achieve what is needed:**

1. **Backend Configuration:**
2. **application.properties:**
3. **Server Port:**

* **server.port=8089:** This property sets the port number for the application to run on. In this case, it’s set to 8089.

1. **Data Source Configuration:**

* **spring.datasource.url=jdbc:mysql://localhost:3306/goteam?createDatabaseIfNotExist=true:** specifies the URL of the database. Here, mysql is being used that includes the hostname, post 3306 (default for mysql), and database name. **?createDatabaseIfNotExist=true** parameter instructs the driver to create the database if it doesn’t exist when connecting.
* **spring.datasource.username=<username>:** username for the database user
* **spring.datasource.password=<password>:** password for the database user
* **spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver:** specifies the JDBC driver class for MySQL.

1. **JPA/Hibernate Configuration:**

* **spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect:** Tells Hibernate which SQL dialect to use,which in this case, is set for MySQL with the InnoDB storage engine.
* **spring.jpa.hibernate.ddl-auto=update:** Configures Hibernate’s schema generation. Common values are **‘none’, ‘update’, ‘create’, ‘create-drop’.** For production, it’s often set to ‘**validate**’ or ‘**none’**.

1. **Logging Configuration:**

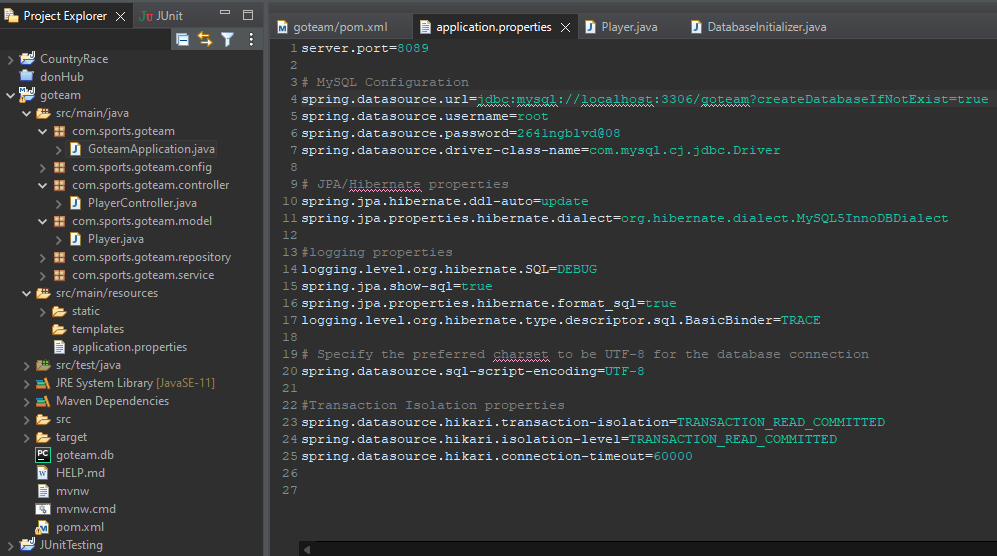
* **logging.level.org.hibernate.SQL=DEBUG**
* **spring.jpa.show-sql=true**
* **spring.jpa.properties.hibernate.format\_sql=true**
* **logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE**

These properties are set to log SQL statements and bind parameters at a detailed level. This is helpful for debugging.

1. **Transaction Isolation Level:**

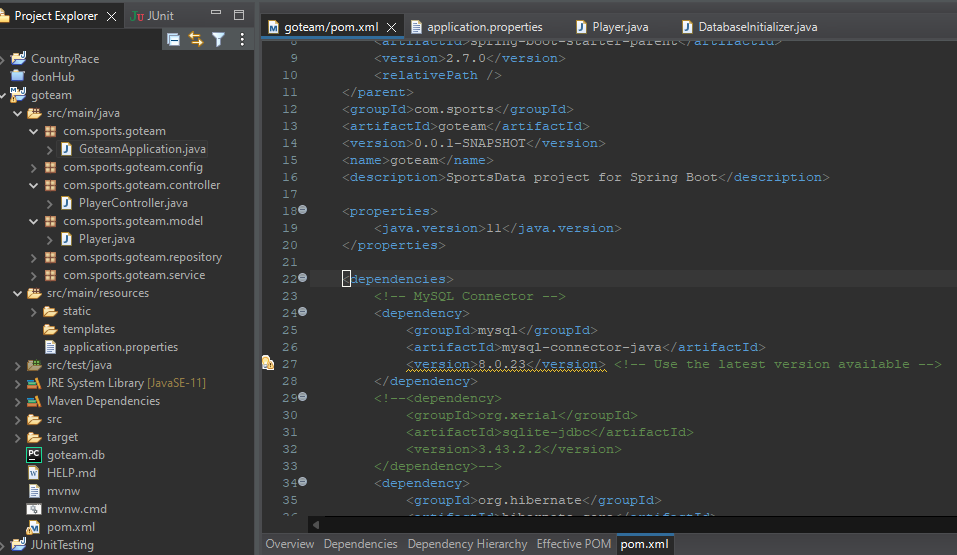
* **spring.datasource.hikari.transaction-isolation=TRANSACTION\_READ\_COMMITTED**
* **spring.datasource.hikari.isolation-level=TRANSACTION\_READ\_COMMITTED**
* **spring.datasource.hikari.connection-timeout=60000**

These properties configure the HikariCP connection pool with a specific transaction isolation level and connection timeout. SQLite supports a different set of isolation levels than some other databases, so it’s important to set this correctly.



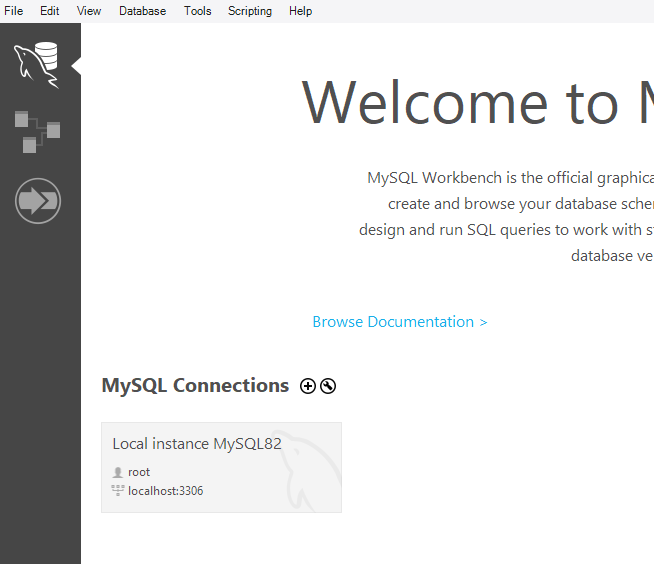
1. **pom.xml:**

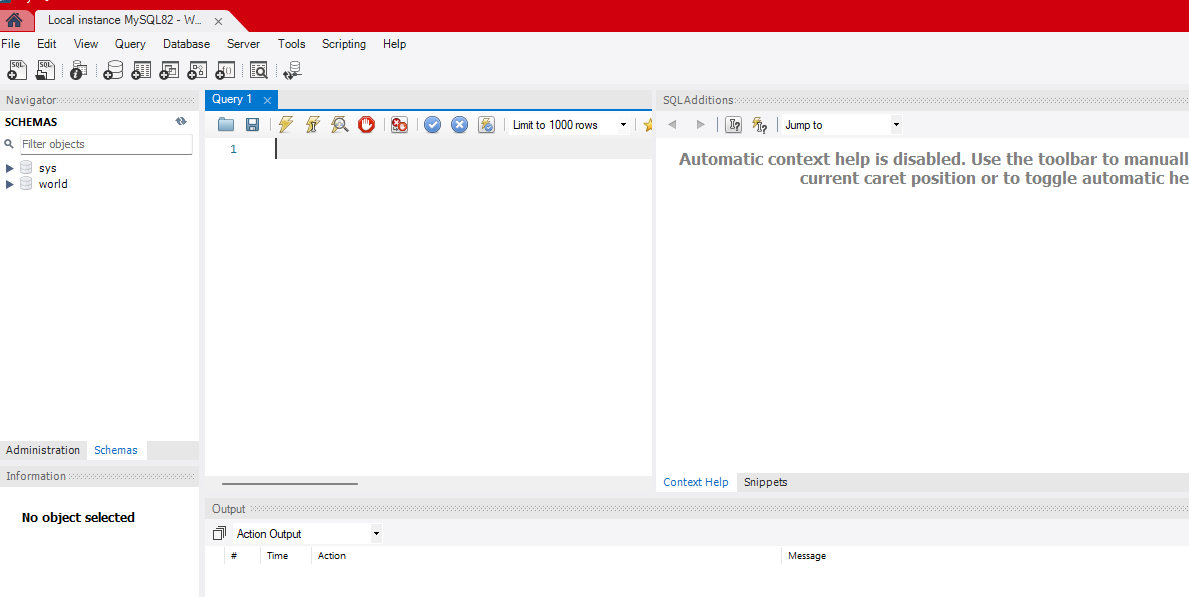
* **mysql:mysql-connector-java:8.0.23** is the MySQL JDBC driver.
* Various dependencies are included for **Spring Boot, Hibernate**, and other libraries.



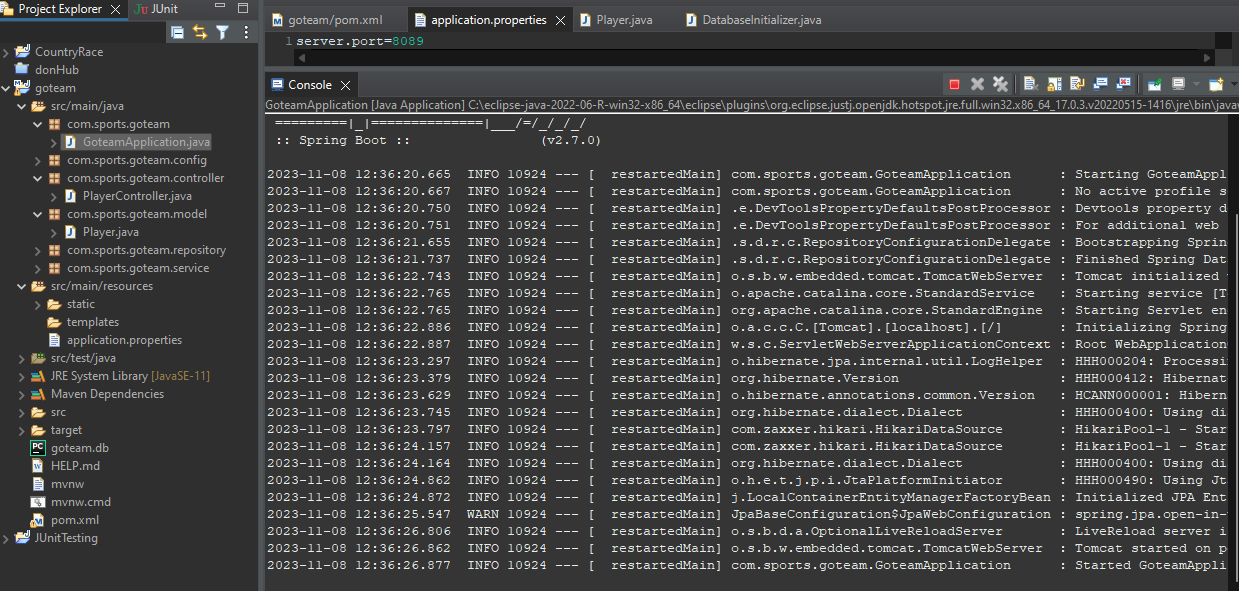
1. **Maven Lifecycle Management:** Throughout the development process, the project was cleaned to remove all files generated by the previous build, installed .jar files in the local repository, and updated to fetch any new dependencies.
2. **Transactional operations:** In the service layer of the application, **@Transactional** annotation is utilized to ensure the consistency and integrity of our database during update and delete operations.
3. **DatabaseInitializer.java:** The **DatabaseInitializer** class in a Spring application is designed to automatically create a MySQL database during the application's startup. It uses the **@PostConstruct** annotation to call the **initialize()** method after the bean's instantiation. This method retrieves the database name from the application's configuration, constructs a SQL statement to create the database if it doesn't already exist, and executes this statement. If there's an error during this process, the class throws a **RuntimeException**, signaling that the database couldn't be created.
4. **Database Modifications:**
5. **MySQL Workbench:**

* The project was initiated by opening MySQL Workbench.
* Under MySQL Connections, select **Local instance MySQL82** that includes username and localhost:<port>

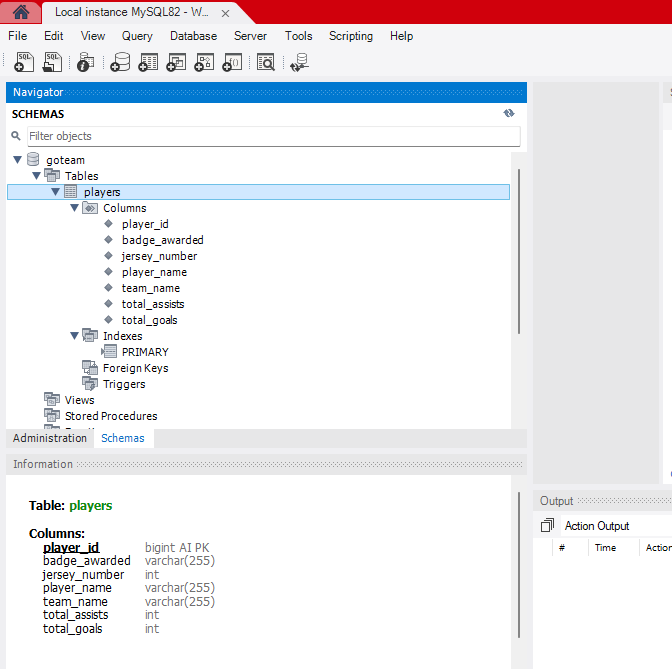
****

****

1. **Application Up and Running:**

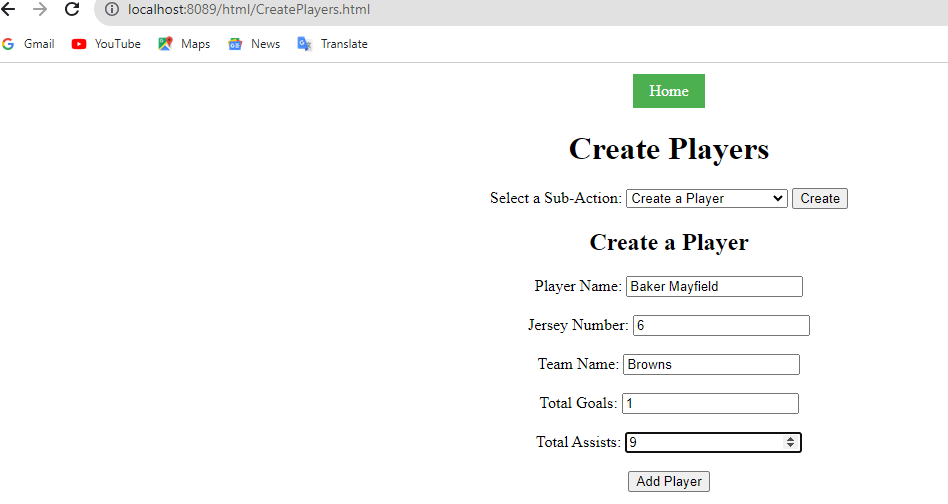
****

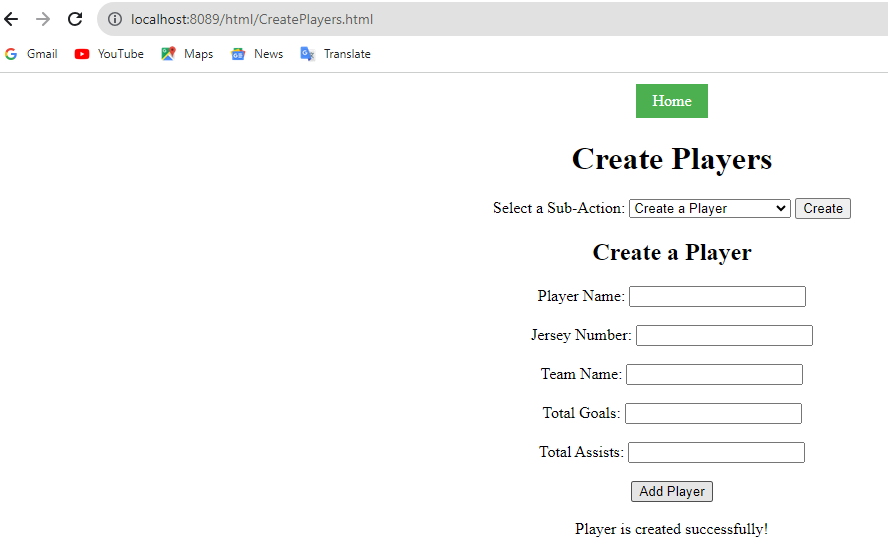
1. **Database=goteam, schema, and table=players creation in MySQL Workbench:**

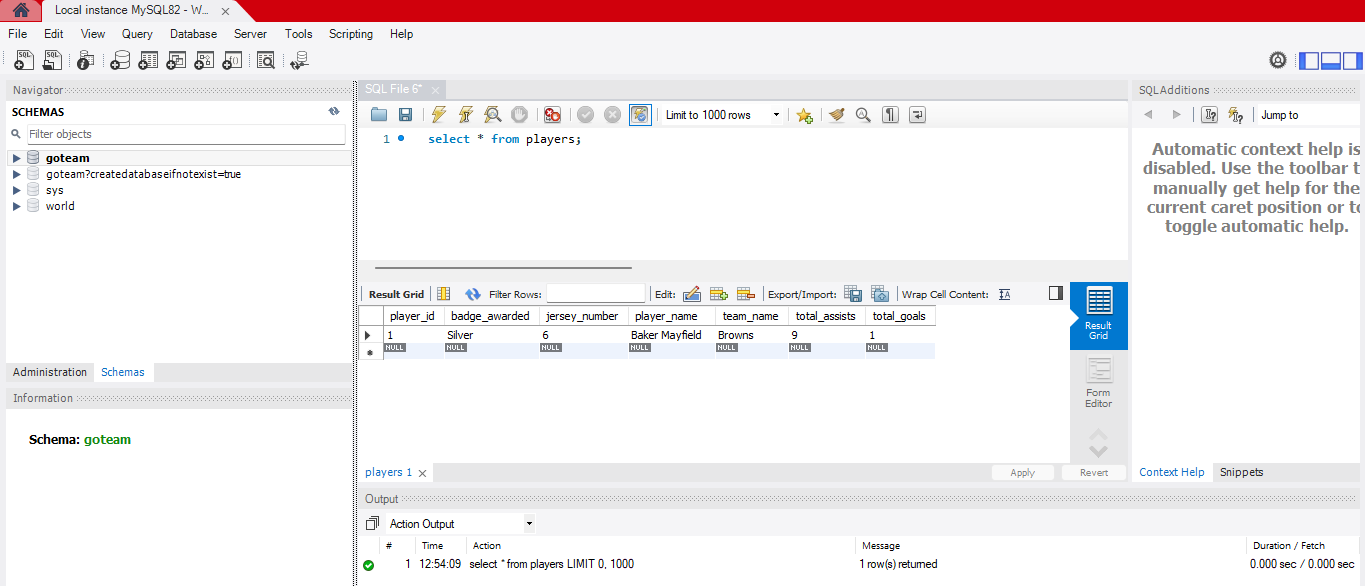
****

1. **Inserting players to the database via UI:**

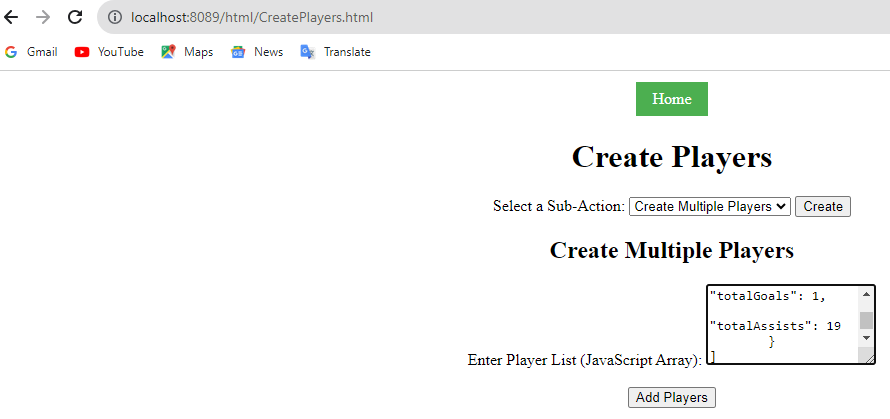
**Inserting single player:**

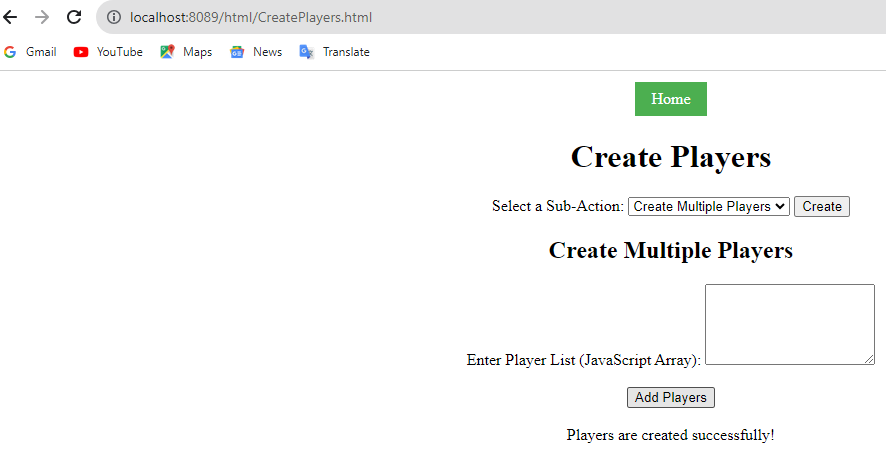
****

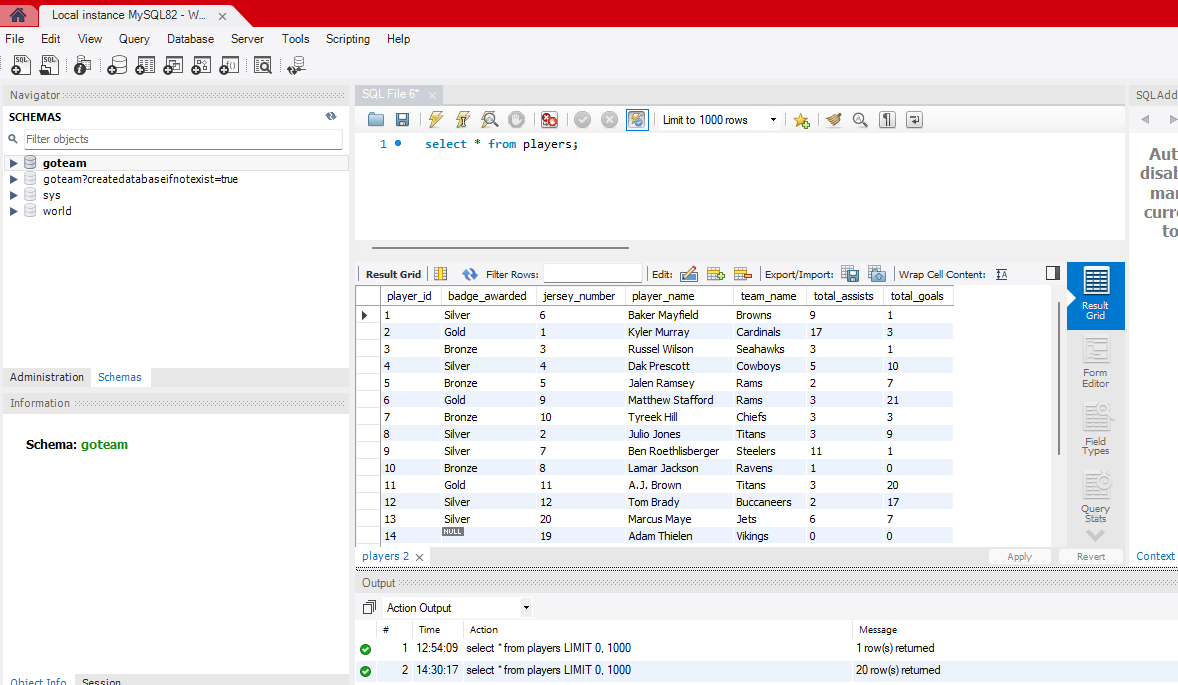
****

****

**Creating multiple players:**

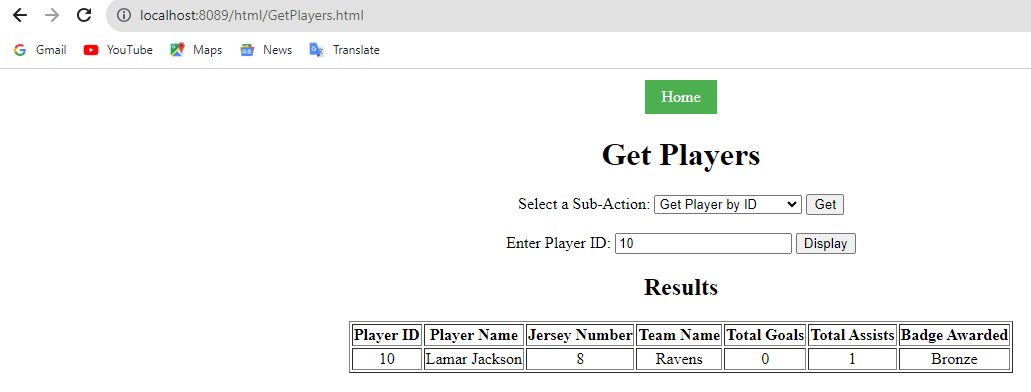
****

****

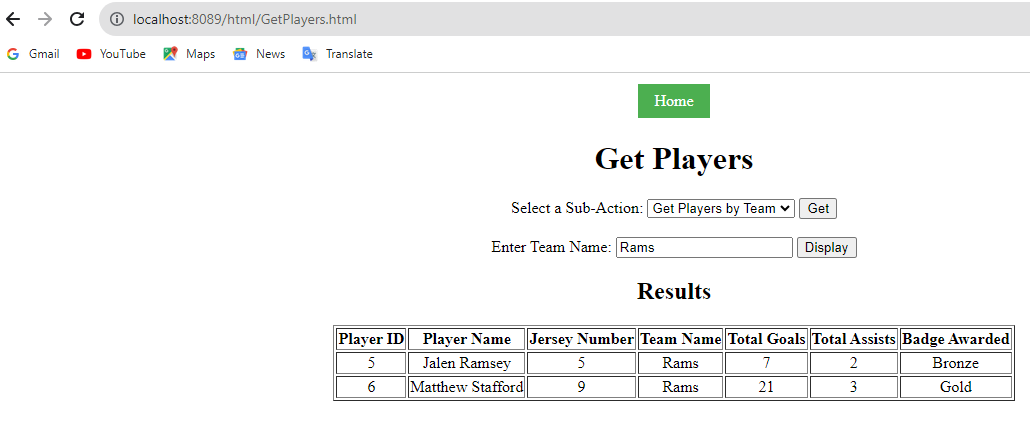
****

1. **Reading players from Sqlite:**

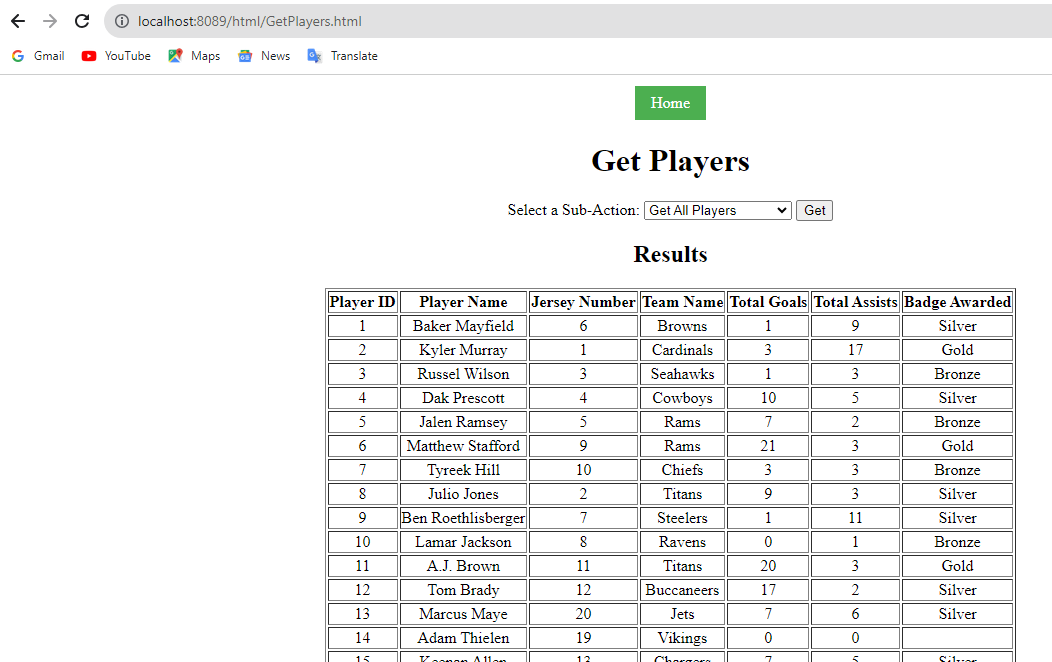
**Reading player by id:**

****

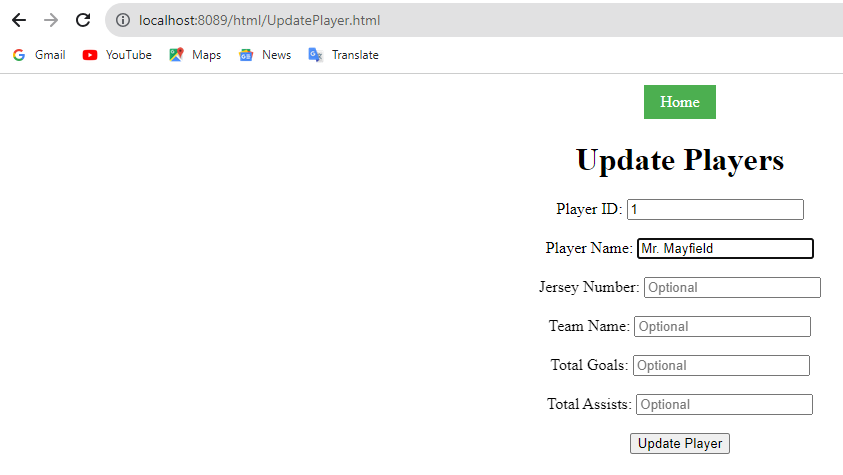
**Reading players by teamname:**

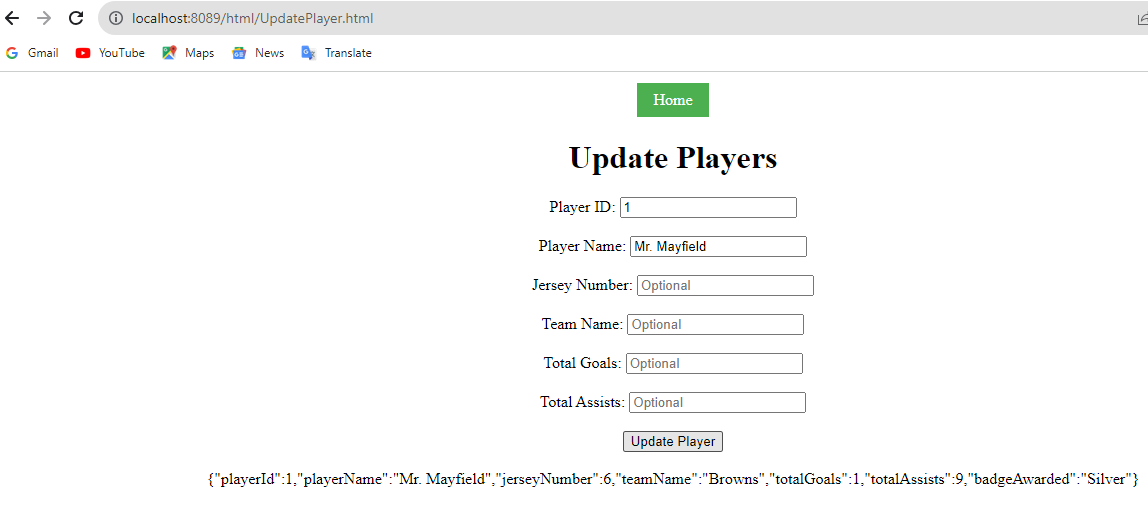
****

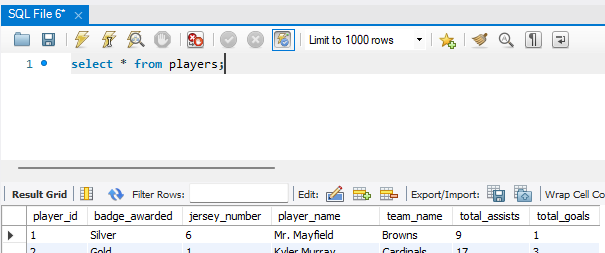
**Reading all players:**

****

1. **Updating Players in the database via UI:**

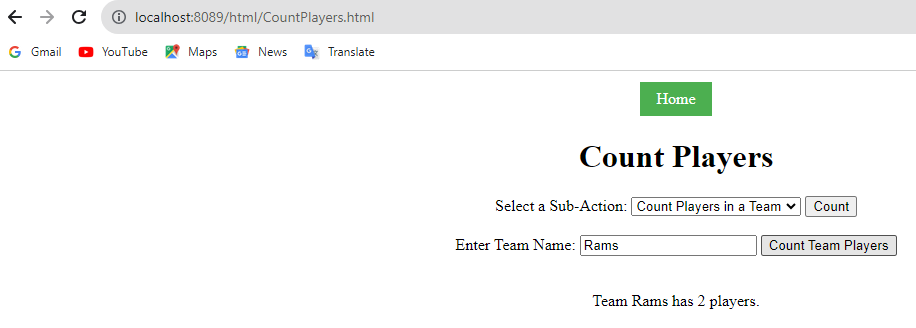
****

****

****

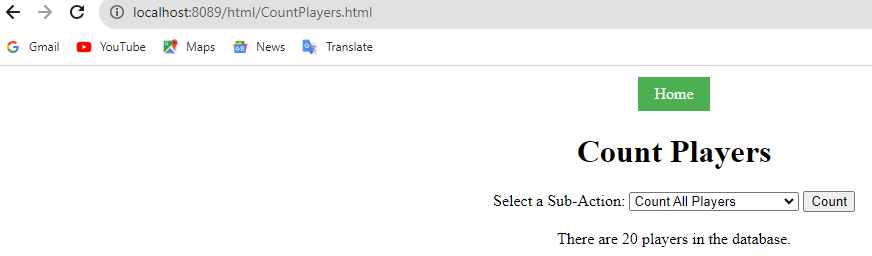
1. **Counting players:**

**Counting players in a team:**

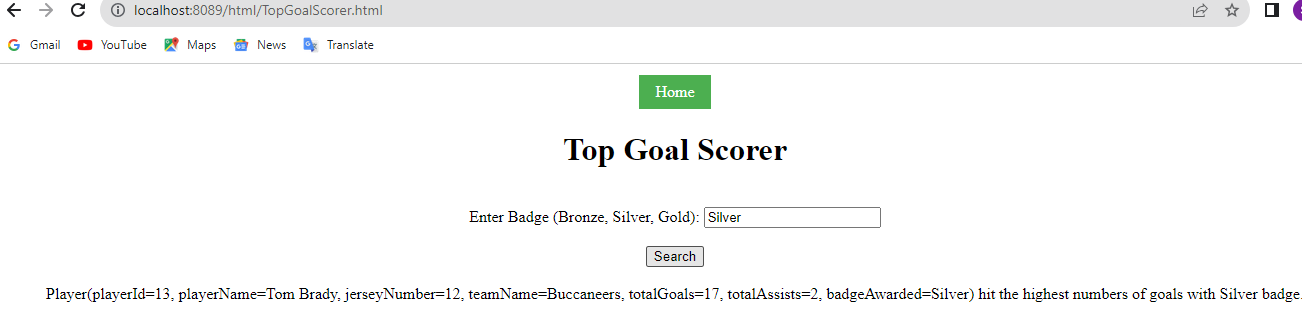
****

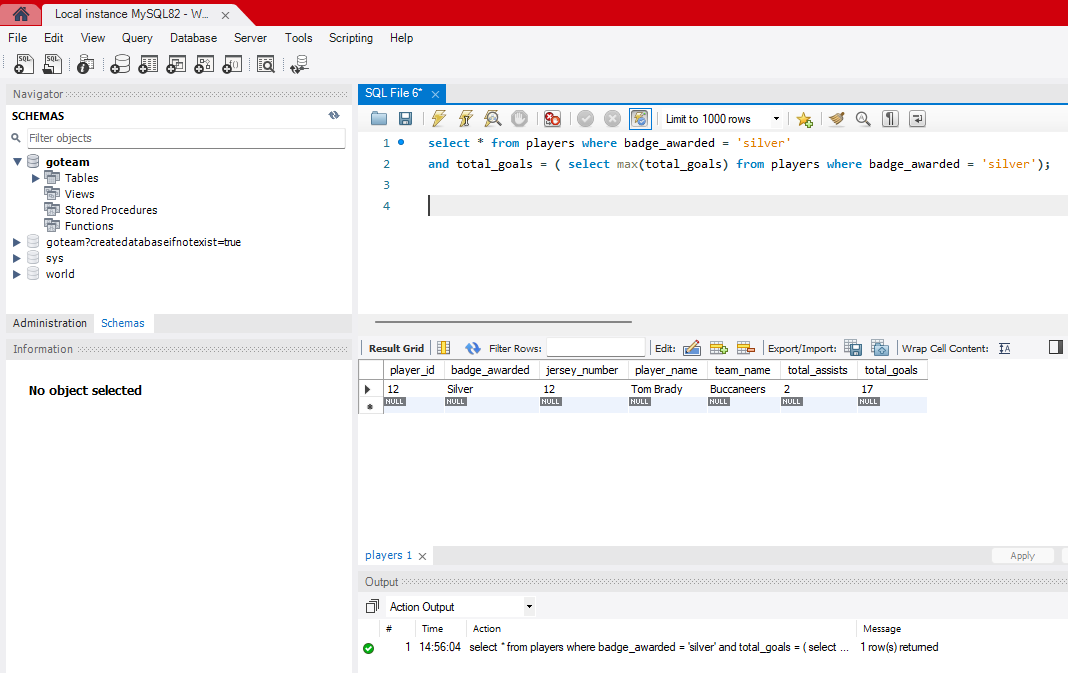
****

**Counting all players:**

****

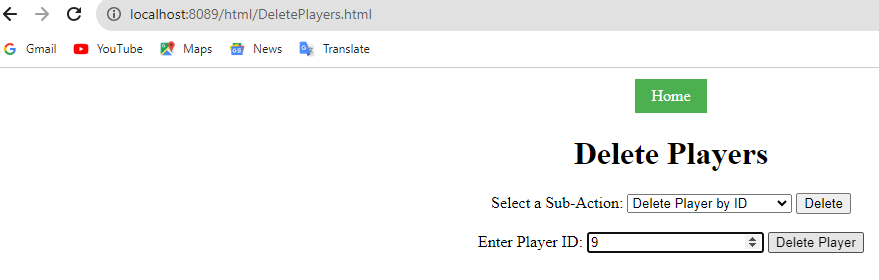
1. **Top Goal Scorer:**

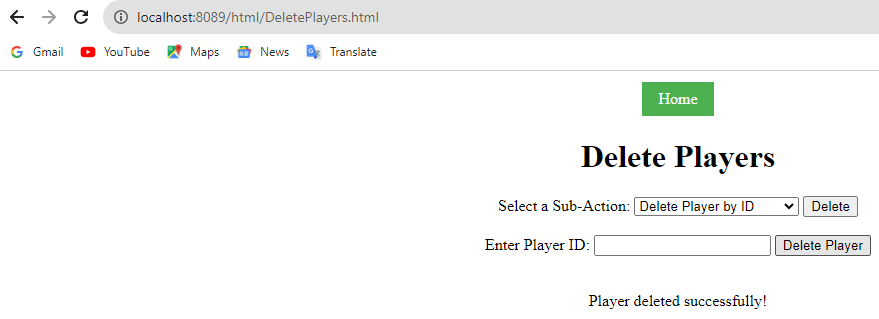
****

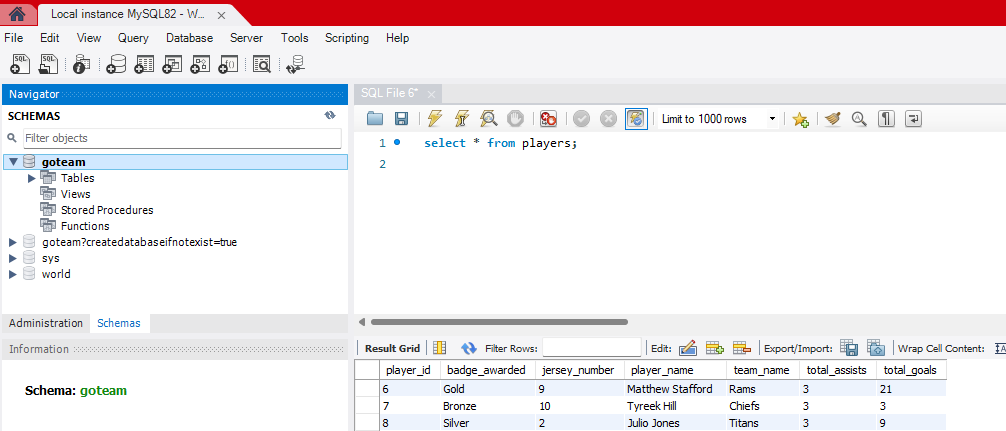
****

1. **Deleting players from Sqlite using UI:**

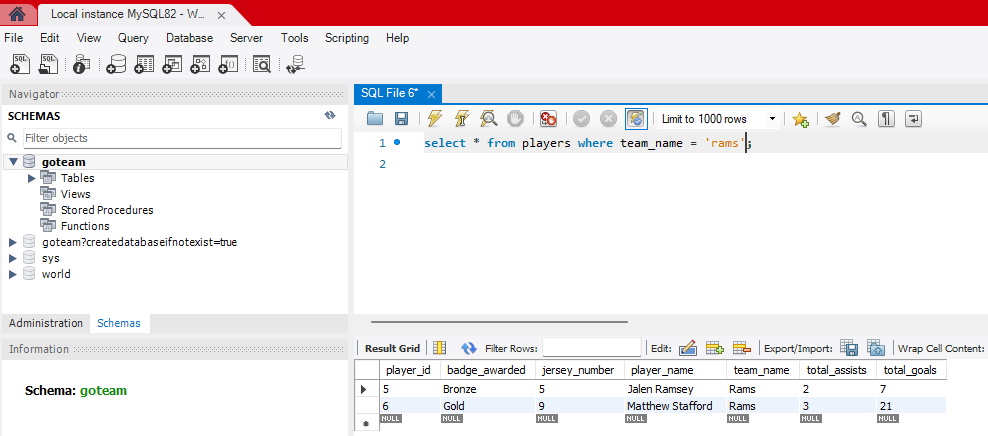
**Deleting player by id:**

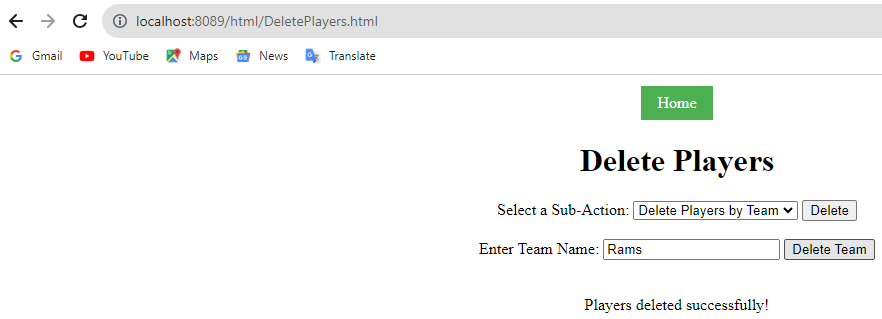
****

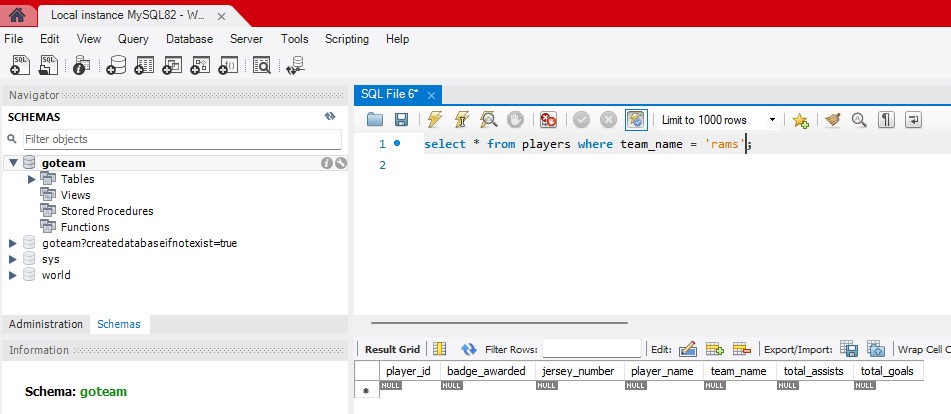
****

****

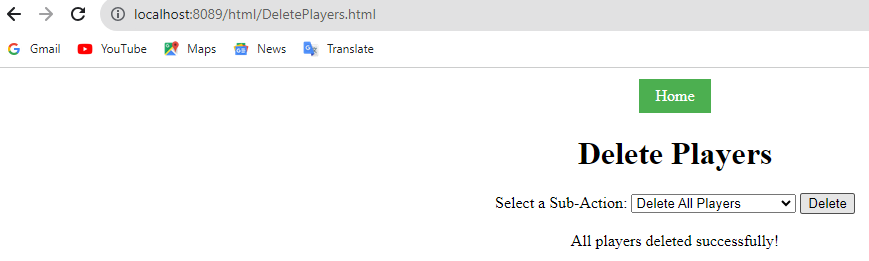
**Deleting players by team:**

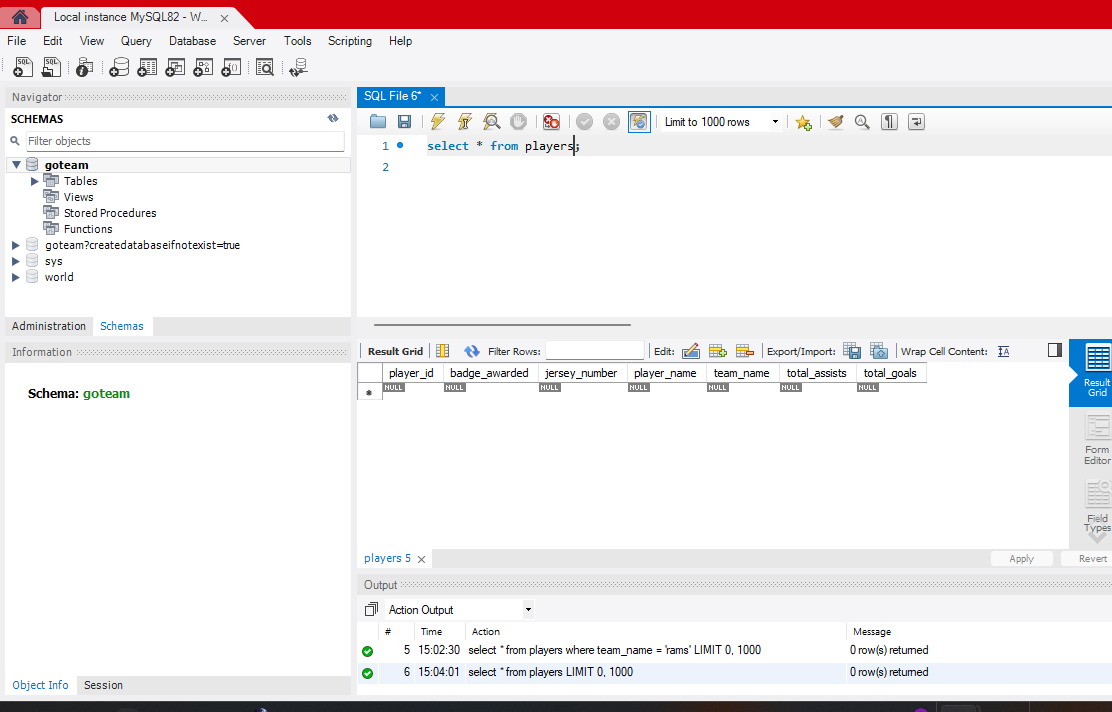
****

****

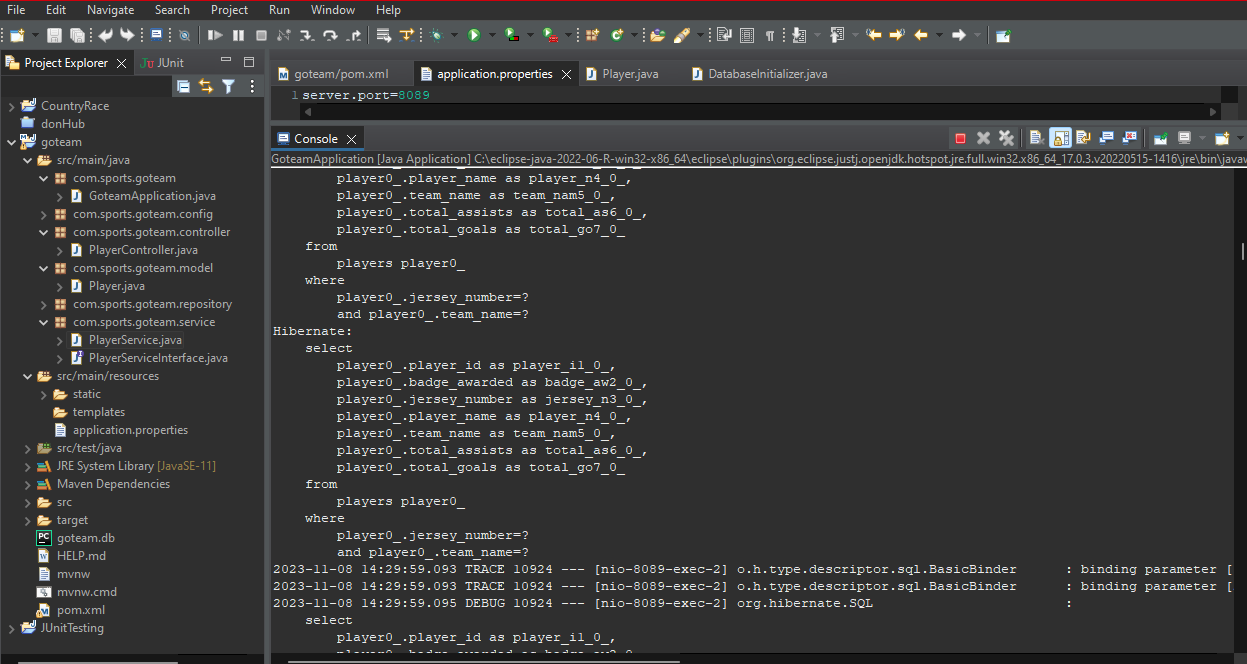
****

**Deleting all players:**

****

****

1. **Logging of console with sql statements after every api request:**

****