Objective

We have a list of online gamers who participate in teams. We need to keep track of some basic information about our players as well as which players are on which teams. There is a list of Players and a list of Teams in the form of existing tables, and there are some stored procedures to access those tables, but there are a lot of problems with the existing design. Your job is to fix the existing structure by modifying, adding and/or removing components as necessary in order to create a solid, well designed database that meets the requirements and deliverables defined below.

Requirements

- Player Table:
  - Each player must have the following information:
    - ID
    - First Name
    - Last Name
    - Date Of Birth
  - Each player may belong to zero or more teams
  - A team must exist before a player can join it
- Team Table:
  - Each team must have a name
  - Each team may have zero or more members
  - Each team with participants may have one and only one leader
- Both Stored Procedures
  - Add an integer return code: 0 indicates success, non-zero indicates an error. If you like you can create different return codes for specific errors, or you can just return any non-zero value to indicate an error.
- SetPlayer Stored Procedure
  - Allow caller to create a new player, edit and existing player or delete an existing player using one stored procedure (SetPlayer)
  - When creating a new player the database should generate a new ID for the caller and return it to the caller – i.e. the caller should not have to pass in an ID for a new user, and the newly generated ID should be returned to the caller in some useful way
  - When creating a new user, an error should be returned if any of the required fields are missing
  - When editing a user, the caller should only have to provide the ID and any fields that he or she wants to change. For example, the caller should be able to pass in just the ID and the LastName value in order to change a player's last name
  - The caller should be able to set / update the teams that a player is associated with in one call, and the stored procedure should accommodate any number of teams in case we add more teams in the future (it is OK to have the caller always pass in a complete list of teams that the player is associated with and replace the entire list)
- GetPlayer Stored Procedure
  - All arguments should be treated as filters. If any argument is passed in, it will be used to filter the result set, if any argument is not passed in, the result set will not be filtered on that field – all arguments should therefore be optional

- It is OK for the @Team argument to only filter on a single team – i.e. it is not necessary to come up with a way to filter on more than one team in a single call

Deliverables

1. Tables
    a. Provide a **Player** table and a **Team** table, as described above
    b. You may modify the existing tables or create new ones
    c. The initial tables must start with the same sample data provided
2. Stored Procedures
    a. Provide a **SetPlayer** stored procedure and a **GetPlayer** stored procedure, as described above
    b. You may modify the existing stored procedures or create new ones
    c. Both stored procedures must provide a return code as described above

Considerations

- Obviously a complete example would include a Set/Get procedure for the Team table but for this exercise it is acceptable to create / modify the Team table by hand
- All of the T-SQL in the provided DDL is written in mixed case, please follow that convention when putting together your solution.
- Please use tabs and not spaces in your final table DDL and stored procedure code.
- Please provide your final solution in a simple text file with a .txt or .sql extension.
- Your solution should be in the form of a script that can be executed all at once (use GO statement separators as needed) to add/edit/delete the tables, stored procedures, etc. that are required to meet the requirements.
- The table structure is intentionally flawed and incomplete, please be sure to add/change/remove the DDL as needed to create well structured tables that meet the basic requirements of primary keys, referential integrity, etc.
- The stored procedures here are intentionally horribly written, please be sure to remove any poorly written constructs / logic and make the procedures as logical, efficient and easy to read as possible. Do not assume that any logic in either of the stored procedures needs to be kept in the final product. The only part of the stored procedure code that you must keep is the formatting – please do your best to make your final code follow the formatting conventions used.

Final Notes

Please provide your solution in the form of a T-SQL script that can be executed on a SQL Server instance all at once (use GO statement separators as needed) to add/edit/delete the tables, stored procedures, etc. that are needed to meet the requirements. We are using SQL Server 2016 so please be sure to only use statements that are supported on that version of SQL Server.

We are looking for a *simple* solution. Please do not add any unnecessary complication to the DDL or stored procedure code or use any components or constructs that are not necessary to complete the evaluation. The ideal solution will be well formatted, uncomplicated and easy to read.

Table Creation and Population DDL

```
create table Player
(
   ID        int,
   LastName  varchar(50),
   FirstName varchar(50),
   DOB       varchar(10),
   Team1     varchar(50),
   Team2     varchar(50),
   Team3     varchar(50)
)

create table Team
(
   ID         int,
   TeamName   varchar(50),
   TeamLeader varchar(100)
)

insert into Player values
(100, 'Smith', 'Sally', '04/18/1995', 'Red', null, null),
(101, 'Jones', 'James', '05/22/1996', 'Blue', 'Red', null),
(102, 'Brown', 'Bobby', '11/18/1995', 'Green', 'Blue', null),
(103, 'Jones', 'Janet', '08/22/1998', 'Blue', 'Green', 'Red'),
(104, 'Magoo', 'Matthew', '12/20/1994', 'Green', null, null),
(105, 'Wayfield', 'Wanda', '05/05/1995', 'Red', null, null)

insert into Team values
(10, 'Red', 'Sally Smith'),
(20, 'Blue', 'Bobby Brown'),
(30, 'Green', 'Matthew Magoo')
```

SetPlayer Stored Procedure

```
create procedure SetPlayer
   @ID        int,
   @LastName  varchar(50),
   @FirstName varchar(50),
   @DOB       varchar(10),
   @Team1     varchar(50),
   @Team2     varchar(50),
   @Team3     varchar(50)
as begin
   set nocount on

   declare @TempID as int
   select  @TempID = ID
   from    Player
   where   ID = @ID

   if @TempID is null begin
      insert
      into   Player
      values (
                @ID,
                @LastName,
                @FirstName,
                @DOB,
                @Team1,
                @Team2,
                @Team3
             )
   end

   if @TempID is not null begin
      update Player
      set    LastName = @LastName,
             FirstName = @FirstName,
             DOB = @DOB,
             Team1 = @Team1,
             Team2 = @Team2,
             Team3 = @Team3
      where  ID = @ID
   end
end
```

GetPlayer Stored Procedure

```
create procedure GetPlayer
   @ID        int,
   @LastName  varchar(50),
   @FirstName varchar(50),
   @Team      varchar(50)
as begin
   set nocount on

   if @ID is not null begin
      select *
      from   Player
      where  ID = @ID
   end else if @LastName is not null begin
      select *
      from   Player
      where  LastName = @LastName
   end else if @FirstName is not null begin
      select *
      from   Player
      where  FirstName = @FirstName
   end else if @Team is not null begin
      select *
      from   Player
      where  Team1 = @Team
      or     Team2 = @Team
      or     Team3 = @Team
   end

end
```