# Assignment 2

In this assignment, groups will be taking an existing database design and expanding it using PL/SQL and various objects to better support a business application. Through the process of understanding the business requirements, the business rules, and how the database will work with the software application, learners will be able to identify ways, or solutions, in which the database design can be extended to be an integral part of the software architecture, and not just a storage and retrieval facility.

Groups will be creating the PL/SQL code that will create the required objects in a database to support the business requirements, and business rules, as well as various objects to support the user interface and basic CRUD operations through a secure parameterized method.

# Submission

Each group will have 3 parts to their assignment submission.

1. A single .SQL file contained all the code required to create the objects required for the assignment. You will only create the objects, and provide the execution code in a non-saved procedure.
2. A users guide to the created objects listing each object created including:
   * Required input parameters (type, size, and meaning)
   * expected outputs,
   * potential error codes,
   * a description of its’ purpose, and
   * an example of non-saved procedural code to execute the given object.
3. An INDIVIDUAL short video (3 to 5 minutes), presenting the solution, and discussing the roles of the objects in a potential software scenario. Each member must create their own video separately.   
     
   The video should include things like:
   * what the objects are, and how they work (not everyone, you only have 5 minutes)
   * how the objects can be used in a software solution?
   * what are the benefits/drawbacks of building these objects in the database as opposed to in the software application?

Expectations for the video:

* + Each student creates their own individual video,
  + Each student must demonstrate their knowledge of the designed objects,
  + Each student must be present through **voice** (webcam optional), so a mic will be required,
  + Any video that goes beyond 5 minutes will only be viewed and marked based on the first 5 minutes
  + Videos can be recorded anyway you find works, however, if no alternative method is available, you can use your MS Teams group channel to hold a meeting, and simply record it while doing the presentation, you can then download the recording and submit it.
  + Videos can be uploaded to YouTube, and just the link provided.
    1. Make sure that videos uploaded to YouTube are marked as **unlisted**.   
       - Private videos will not be marked as I will not be able to see them.   
       - Public videos will be marked as if you made your solutions available to other groups (copying academic integrity).

# Tasks

The assignment is broken into a few tasks and are based on the SportLeagues database that has already been provided to you.

1. For each table in (Players, Teams, Rosters) create Stored Procedures to cover the 4 basic CRUD tasks.
   1. INSERT a new record and if the PK using autonumber, the SP returns the new PK,
   2. UPDATE an existing record given the PK value,
   3. DELETE an existing record given the PK value, and
   4. SELECT return all fields in a single row from a table given a PK value.

* Name the SPs using the following guide: spTableNameMethod (example spPlayersInsert)
* Do not use DBMS\_OUTPUT in the procedures in any way. If you use it for debugging purposes, make sure it is commented out in the final submission.
* All SPs must have appropriate exception handling specific to the method and table.
* Use error codes of the same type and size of the PK to return values that can be clearly determined to indicate an error (example: -1 might indicate no record was found). These should be consistent values across all SPs such that only a single table of error codes is required in the documentation.

1. For each table in (Players, Teams, Rosters), create an additional Stored Procedure, called spTableNameSelectAll that outputs the contents of the table to the script window (using DBMS\_OUTPUT) for the standard SELECT \* FROM <tablename> statement.
2. Repeat Step 2 but returning the table in the output of the SP. Use a non-saved procedure to show receiving the data and outputting it to the script window.   
   *NOTE: Some research will be required here as we did not cover this in class.*
3. Create a view which stores the “players on teams” information, called vwPlayerRosters which includes all fields from players, rosters, and teams in a single output table. You only need to include records that have exact matches.
4. Using the vwPlayerRosters view, create an SP, named spTeamRosterByID, that outputs, the team rosters, with names, for a team given a specific input parameter of teamID
5. Repeat task 4, by creating another similar stored procedure, named spTeamRosterByName, that receives a string parameter and returns the team roster, with names, for a team found through a search string. The entered parameter may be any part of the name.
6. Create a view that returns the number of players currently registered on each team, called vwTeamsNumPlayers.
7. Using vwTeamsNumPlayers create a user defined function, that given the team PK, will return the number of players currently registered, called fncNumPlayersByTeamID.
8. Create a view, called vwSchedule, that shows all games, but includes the written names for teams and locations, in addition to the PK/FK values. Do not worry about division here.
9. Create a stored procedure, spSchedUpcomingGames, using DBMS\_OUTPUT, that displays the games to be played in the next n days, where n is an input parameter. Make sure your code will work on any day of the year.
10. Create a stored procedure, spSchedPastGames, using DBMS\_OUTPUT, that displays the games that have been played in the past n days, where n is an input parameter. Make sure your code will work on any day of the year.
11. Using the Standings calculation demo code provided earlier in the semester, create a Stored Procedure, named spRunStandings, that replaces a temporary static table, named tempStandings, with the output of the SELECT code provided.
12. Following up with Step 12, create a Trigger in the system to automate the execution of the spRunStandings SP when any row in the games table is updated. Essentially meaning that software can run SELECT \* FROM tempStandings; and always have up to date standings.
13. Each group must be creative and come up with an object (SP, UDF, or potentially trigger), of your own choosing, that will be built in the database to help support the same ideals of the above objects.