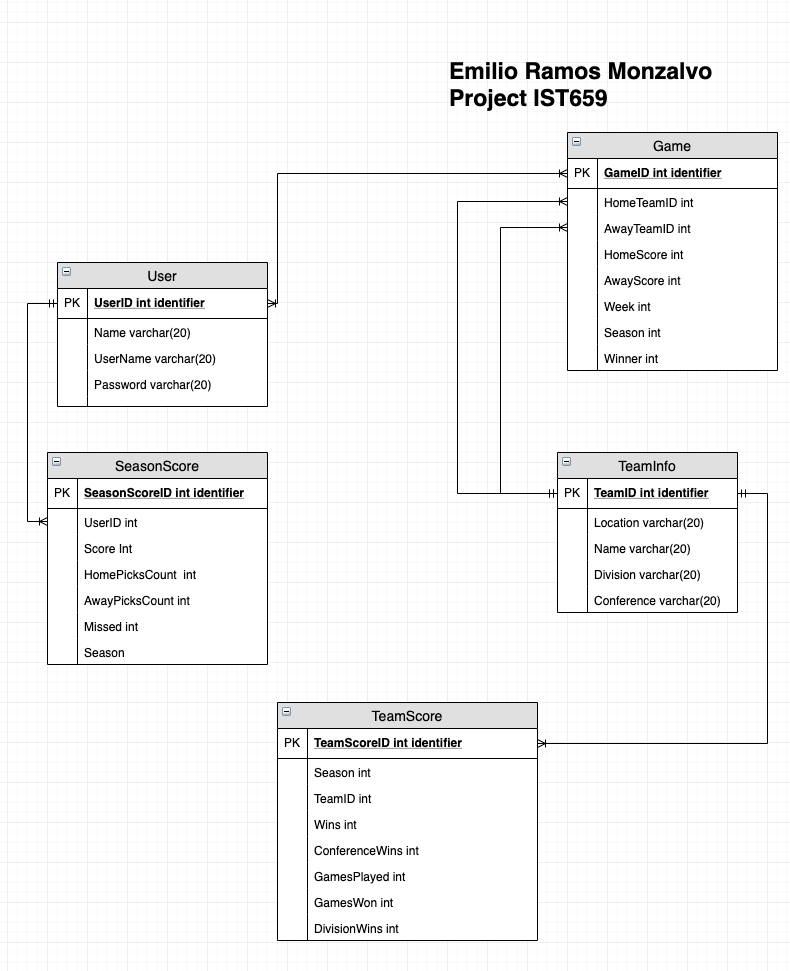
**IST 659 Project Deliverable 1**

**Summary 1:**

Since I was a young kid, my dad has always been a big fan of the NFL. He is such a big fan that every year he makes the whole family pick the winners for each game. Each week he lists all of the games and we have to pick a winner as you can see in the picture below. Then, at the end of the season he tallies all of our correct picks and one of the four of us is crowned the winner. He has data like this for the past 10 years in an excel file where he also keeps a bunch of statistics. I eventually want to create a web application where we can keep track of all of our picks and the games’ scores. This will require a well-made Relational Database that includes the divisions of the teams, scores, and will be able to calculate playoff contenders by itself. Then, I would also like to post simple statistics like which team is my most accurate pick each year or throughout the whole lifetime.



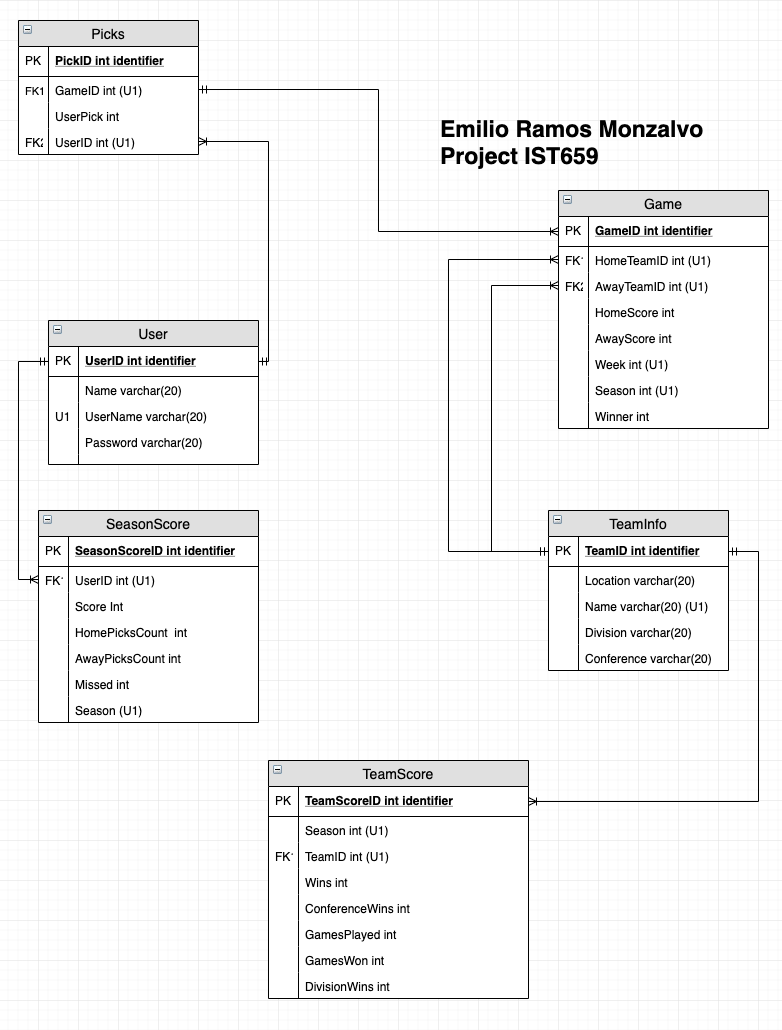
**Conceptual Model:**



Glossary:

* User: Used for login and user information
  + UserID: identifier for users
  + Name:
  + username: used for login information
  + Password: used for login information
* SeasonScore: used to keep trak of the users statistics on how many games he has predicted correctly
  + seasonScoreID: Unique identifier for the entity
  + UserID: used to check who’s score this is
  + Score: How many the user has gotten correct
  + HomePicksCount: How many teams he has that were playing a home
  + AwayPicksCount: How many teams he has that were playing away
  + Missed: How many games he has predicted incorrectly
  + Season: What season we are on
* Game: Keeps track of the teams schedule throughout the season
  + gameID: Unique identifier for the entity
  + HomeTeamID
  + AwayTeamID
  + HomeTeamScore
  + AwayTeamScore
  + Week: what week is the game on
  + Season: what season is the game on
  + Winner: who won the game
* TeamInfo: Used to keep track of the teams conference and division for playoff calculations
  + teamID: Unique identifier for the entity
  + Location
  + Name
  + Division
  + Conference
* TeamScore: Used to keep track how a team is doing in season
  + TeamScoreID: unique identifier for the entity
  + teamID: unique team id
  + season: Unique year in which the score is reflected on
  + GamesWon
  + GamesPlayed
  + ConferenceWins
  + DivisionWins

**Normalized Model:**



I added the picks entity in order to get rid of the many-to-many relationship between the Game and User entity. This way it is much easier to track which user picked what team in each game.

**IST 659 Project Deliverable 2**

**Physical Database Design:**

/\*

Author : Emilio Ramos Monzalvo

Course : IST659 M400

Term : September, 2019

\*/

-- Creating the User table

CREATE TABLE Users (

-- Columns for the User table

UserID int identity,

FirstName varchar(20) not null,

LastName varchar(20) not null,

UserName varchar(20) not null,

Password varchar(50) not null

-- Constrainst on the User Table

CONSTRAINT PK\_Users PRIMARY KEY (UserID),

CONSTRAINT U1\_Users UNIQUE (UserName)

)

-- End Creating the User table

-- Creating the Seasons table

CREATE TABLE Seasons (

-- Columns for the Seasons table

SeasonID int identity,

Season int not null

-- Constrainst on the User Table

CONSTRAINT PK\_Seasons PRIMARY KEY (SeasonID),

CONSTRAINT U1\_Seasons UNIQUE (Season)

)

-- End Creating the Season table

-- Creating the SeasonScore table

CREATE TABLE SeasonScore (

-- Columns for the SeasonScore table

SeasonScoreID int identity,

UserID int not null,

Score int not null DEFAULT 0,

HomePicksCount int not null DEFAULT 0,

AwayPicksCount int not null DEFAULT 0,

MissedCount int not null DEFAULT 0,

SeasonID int not null,

-- Constrainst on the SeasonScore Table

CONSTRAINT PK\_SeasonScore PRIMARY KEY (SeasonScoreID),

CONSTRAINT U1\_SeasonScore UNIQUE (UserID, SeasonID),

CONSTRAINT FK1\_SeasonScore FOREIGN KEY (UserID) REFERENCES Users(UserID),

CONSTRAINT FK2\_SeasonScore FOREIGN KEY (SeasonID) REFERENCES Seasons(SeasonID)

)

-- End Creating the SeasonScore table

-- Creating the TeamInfo table

CREATE TABLE TeamInfo (

-- Columns for the TeamInfo table

TeamID int identity,

LocationOfTeam varchar(20) not null,

TeamName varchar(20) not null,

Division varchar(20) not null,

Conference varchar(20) not null,

-- Constrainst on the TeamInfo Table

CONSTRAINT PK\_TeamInfo PRIMARY KEY (TeamID),

CONSTRAINT U1\_TeamInfo UNIQUE (TeamName)

)

-- End Creating the TeamInfo table

-- Creating the Game table

CREATE TABLE Game (

-- Columns for the Game table

GameID int identity,

HomeTeamID int not null,

AwayTeamID int not null,

HomeScore int not null DEFAULT 0,

AwayScore int not null DEFAULT 0,

WeekOfSeason int not null,

SeasonID int not null,

Winner int, -- 1 = Home, 0 = Away Team

-- Constrainst on the Game Table

CONSTRAINT PK\_Game PRIMARY KEY (GameID),

CONSTRAINT U1\_Game UNIQUE (HomeTeamID, AwayTeamID, WeekOfSeason, SeasonID),

CONSTRAINT FK1\_Game FOREIGN KEY (HomeTeamID) REFERENCES TeamInfo(TeamID),

CONSTRAINT FK2\_Game FOREIGN KEY (AwayTeamID) REFERENCES TeamInfo(TeamID),

CONSTRAINT FK3\_Game FOREIGN KEY (SeasonID) REFERENCES Seasons(SeasonID)

)

-- End Creating the Game table

-- Creating the Picks table

CREATE TABLE Picks (

-- Columns for the Picks table

PickID int identity,

GameID int not null,

UserPick int, -- 1 = Home, 0 = Away Team

UserID int not null,

-- Constrainst on the Picks Table

CONSTRAINT PK\_Picks PRIMARY KEY (PickID),

CONSTRAINT U1\_Picks UNIQUE (GameID, UserID),

CONSTRAINT FK1\_Picks FOREIGN KEY (UserID) REFERENCES Users(UserID),

CONSTRAINT FK2\_Picks FOREIGN KEY (GameID) REFERENCES Game(GameID)

)

-- End Creating the Picks table

-- Creating the TeamScore table

CREATE TABLE TeamScore (

-- Columns for the TeamScore table

TeamScoreID int identity,

SeasonID int not null,

TeamID int not null,

Wins int not null DEFAULT 0,

ConferenceWins int not null DEFAULT 0,

GamesPlayed int not null DEFAULT 0,

GamesWon int not null DEFAULT 0,

DivisionWIns int not null DEFAULT 0,

-- Constrainst on the TeamScore Table

CONSTRAINT PK\_TeamScore PRIMARY KEY (TeamScoreID),

CONSTRAINT U1\_TeamScore UNIQUE (SeasonID, TeamID),

CONSTRAINT FK1\_TeamScore FOREIGN KEY (TeamID) REFERENCES TeamInfo(TeamID),

CONSTRAINT FK2\_TeamScore FOREIGN KEY (SeasonID) REFERENCES Seasons(SeasonID)

)

-- End Creating the TeamScore table

**Data Creation:**

There should only be two INSERT statements into a Table that do not have any dependencies on any other Table. This includes the User and the TeamInfo Table. Then there should only be two more insert statements. The first one is the ‘New Season’ Insert Procedure. This procedure is used once a season. It creates a row in the SeasonScore Table for each User, and it also creates a row in the TeamScore for every Team in the TeamInfo Table. The second insert procedure is done for every new game. This procedure creates a row in the Picks Table for every User in the User Table.

When adding data to the Users Table, the UserID is computed automatically as shows below.

-- Adding Data to the Users table

INSERT INTO Users (FirstName, LastName, UserName, Password)

VALUES

('Emilio', 'Ramos', 'eramos', 'password'),

('Hector', 'Ramos', 'hramos', 'password'),

('Mama', 'Ramos', 'mramos', 'password'),

('Papa', 'Ramos', 'pramos', 'password')

SELECT \* FROM Users

A screenshot of a computer

Description automatically generated

When adding data to the TeamInfo Table, the TeamID is computed automatically as shows below.

-- Adding Data to the TeamInfo table

INSERT INTO TeamInfo (TeamName, LocationOfTeam, Division, Conference)

VALUES

('Packers', 'Green Bay', 'North', 'NFC'),

('Cowboys', 'Dallas', 'East', 'NFC'),

('Chiefs', 'Kansas City', 'West', 'AFC'),

('Patriots', 'New England', 'East', 'AFC'),

('49rs', 'San Francisco', 'West', 'NFC'),

('Saints', 'New Orleans', 'South', 'NFC'),

('Titans', 'Tennesse', 'South', 'AFC'),

('Steelers', 'Pittsburg', 'North', 'AFC')

SELECT \* FROM TeamInfo

A close up of a street

Description automatically generated

-- Procedure To Create New Season

CREATE PROCEDURE newSeason(@season int)

AS

BEGIN

BEGIN TRANSACTION

BEGIN TRY

INSERT INTO Seasons (Season) VALUES (@season);

DECLARE @SeasonID int;

SELECT @SeasonID = SeasonID FROM Seasons WHERE Season = @season;

INSERT INTO SeasonScore (UserID, SeasonID)

SELECT UserID, @SeasonID FROM Users;

INSERT INTO TeamScore (TeamID, SeasonID)

SELECT TeamID, @SeasonID FROM TeamInfo;

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

END CATCH

END

EXEC newSeason 2019

SELECT \* FROM Seasons

SELECT \* FROM SeasonScore

SELECT \* FROM TeamScore

A screenshot of a cell phone

Description automatically generated

-- Procedure to Create New Game

CREATE PROCEDURE newGame(@HomeTeam varchar(20), @AwayTeam varchar(20), @Week int, @season int)

AS

BEGIN

BEGIN TRANSACTION

BEGIN TRY

DECLARE @SeasonID int;

DECLARE @HomeTeamID int;

DECLARE @AwayTeamID int;

SELECT @SeasonID = SeasonID FROM Seasons WHERE Season = @season;

SELECT @HomeTeamID = TeamID FROM TeamInfo WHERE TeamName = @HomeTeam;

SELECT @AwayTeamID = TeamID FROM TeamInfo WHERE TeamName = @AwayTeam;

INSERT INTO Game (HomeTeamID, AwayTeamID, SeasonID, WeekOfSeason)

VALUES (@HomeTeamID, @AwayTeamID, @SeasonID, @Week);

DECLARE @GameID int;

SELECT @GameID = GameID FROM Game

WHERE HomeTeamID = @HomeTeamID AND AwayTeamID = @AwayTeamID AND WeekOfSeason = @Week AND SeasonID = @SeasonID;

INSERT INTO Picks (UserID, GameID)

SELECT UserID, @GameID FROM Users;

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

END CATCH

END

EXEC newGame 'Packers', 'Cowboys', 1, 2019

EXEC newGame 'Chiefs', 'Patriots', 1, 2019

EXEC newGame '49rs', 'Saints', 1, 2019

EXEC newGame 'Titans', 'Steelers', 1, 2019

SELECT \* FROM Game;

SELECT \* FROM Picks;

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Data Manipulation:**

For this section we need to create two UPDATE statements for when a Game finishes and for when a User makes his pick for a Game. In the first case we need to check the Users picks for that Game and update the SeasonScore Table accordingly. In the second case we just need to update the empty value for pick in the Picks Table. We also need to remember that 1 in the column UserPick means that the User picked the Home Team to win, and if it is 0, then the User picked the Away Team to win.

1. UserPick:

-- Procedure To Create New Season

CREATE PROCEDURE pickGame(@HomeTeam varchar(20), @AwayTeam varchar(20), @season int, @Week int, @UserName varchar(20), @TeamPick varchar(20))

AS

BEGIN

BEGIN TRANSACTION

BEGIN TRY

DECLARE @SeasonID int;

DECLARE @HomeTeamID int;

DECLARE @AwayTeamID int;

DECLARE @UserID int;

DECLARE @UserPick int;

DECLARE @GameID int;

SELECT @SeasonID = SeasonID FROM Seasons WHERE Season = @season;

SELECT @HomeTeamID = TeamID FROM TeamInfo WHERE TeamName = @HomeTeam;

SELECT @AwayTeamID = TeamID FROM TeamInfo WHERE TeamName = @AwayTeam;

SELECT @UserID = UserID FROM Users WHERE UserName = @UserName;

SELECT @GameID = GameID FROM Game

WHERE HomeTeamID = @HomeTeamID AND AwayTeamID = @AwayTeamID AND WeekOfSeason = @Week AND SeasonID = @SeasonID

IF @TeamPick = @HomeTeam

SELECT @UserPick = 1

ELSE IF @TeamPick = @AwayTeam

SELECT @UserPick = 0

ELSE

ROLLBACK TRANSACTION

UPDATE Picks

SET UserPick = @UserPick

WHERE UserID = @UserID AND GameID = @GameID

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

END CATCH

END

EXEC pickGame 'Packers', 'Cowboys', 2019, 1, 'eramos', 'Packers'

EXEC pickGame 'Packers', 'Cowboys', 2019, 1, 'hramos', 'Packers'

EXEC pickGame 'Packers', 'Cowboys', 2019, 1, 'pramos', 'Cowboys'

EXEC pickGame 'Packers', 'Cowboys', 2019, 1, 'mramos', 'Packers'

EXEC pickGame 'Chiefs', 'Patriots', 2019, 1, 'eramos', 'Patriots'

EXEC pickGame 'Chiefs', 'Patriots', 2019, 1, 'hramos', 'Chiefs'

EXEC pickGame 'Chiefs', 'Patriots', 2019, 1, 'pramos', 'Chiefs'

EXEC pickGame 'Chiefs', 'Patriots', 2019, 1, 'mramos', 'Chiefs'

EXEC pickGame '49rs', 'Saints', 2019, 1, 'eramos', '49rs'

EXEC pickGame '49rs', 'Saints', 2019, 1, 'hramos', '49rs'

EXEC pickGame '49rs', 'Saints', 2019, 1, 'pramos', 'Saints'

EXEC pickGame '49rs', 'Saints', 2019, 1, 'mramos', 'Saints'

EXEC pickGame 'Titans', 'Steelers', 2019, 1, 'eramos', 'Steelers'

EXEC pickGame 'Titans', 'Steelers', 2019, 1, 'hramos', 'Steelers'

EXEC pickGame 'Titans', 'Steelers', 2019, 1, 'pramos', 'Steelers'

EXEC pickGame 'Titans', 'Steelers', 2019, 1, 'mramos', 'Steelers'

A screenshot of a cell phone

Description automatically generated

1. SeasonScore:

I had to create a separate procedure to update the SeasonScore Table for every User.

-- Procedure To Update SeasonScore Table After Game Update

ALTER PROCEDURE UpdateSeasonScore(@UserID int, @Season int, @Pick int, @Winner int)

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION

DECLARE @Correct int;

DECLARE @SeasonID int;

SELECT @SeasonID = SeasonID FROM Seasons WHERE Season = @Season;

IF @Pick = @Winner

SELECT @Correct = 1

ELSE

SELECT @Correct = 0

PRINT(@Correct)

PRINT(@Pick)

UPDATE SeasonScore

SET Score = Score + @Correct, HomePicksCount = HomePicksCount + @Pick,

AwayPicksCount = AwayPicksCount + (1 - @Pick), MissedCount = MissedCount + (1 - @Correct)

WHERE SeasonID = @SeasonID AND UserID = @UserID

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

END CATCH

END

-- Procedure To Create New Season

ALTER PROCEDURE GameScore(@HomeTeam varchar(20), @AwayTeam varchar(20), @season int, @Week int, @HomeScore int, @AwayScore int)

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION

DECLARE @SeasonID int;

DECLARE @HomeTeamID int;

DECLARE @AwayTeamID int;

DECLARE @Winner int;

DECLARE @GameID int;

DECLARE @HomeTeamConf varchar(20);

DECLARE @HomeTeamDiv varchar(20);

DECLARE @AwayTeamConf varchar(20);

DECLARE @AwayTeamDiv varchar(20);

DECLARE @HomeTeamConfWin int;

DECLARE @AwayTeamConfWin int;

DECLARE @HomeTeamDivWin int;

DECLARE @AwayTeamDivWin int;

SELECT @SeasonID = SeasonID FROM Seasons WHERE Season = @season;

SELECT @HomeTeamID = TeamID, @HomeTeamConf = Conference, @HomeTeamDiv = Division FROM TeamInfo WHERE TeamName = @HomeTeam;

SELECT @AwayTeamID = TeamID, @AwayTeamConf = Conference, @AwayTeamDiv = Division FROM TeamInfo WHERE TeamName = @AwayTeam;

SELECT @GameID = GameID FROM Game

WHERE HomeTeamID = @HomeTeamID AND AwayTeamID = @AwayTeamID AND WeekOfSeason = @Week AND SeasonID = @SeasonID

IF @HomeScore > @AwayScore

BEGIN

SELECT @Winner = 1

IF @HomeTeamDiv = @AwayTeamDiv

BEGIN

SELECT @HomeTeamDivWin = 1

SELECT @AwayTeamConfWin = 0

SELECT @HomeTeamConfWin = 1

SELECT @AwayTeamDivWin = 0

END

ELSE IF @HomeTeamConf = @AwayTeamConf

BEGIN

SELECT @HomeTeamDivWin = 0

SELECT @AwayTeamConfWin = 0

SELECT @HomeTeamConfWin = 1

SELECT @AwayTeamDivWin = 0

END

ELSE

SELECT @HomeTeamDivWin = 0

SELECT @AwayTeamConfWin = 0

SELECT @AwayTeamConfWin = 0

SELECT @AwayTeamDivWin = 0

UPDATE TeamScore

SET Wins = Wins + 1, ConferenceWins = ConferenceWins + @HomeTeamConfWin,

GamesPlayed += 1, GamesWon += 1, DivisionWins = DivisionWins + @HomeTeamDivWin

WHERE

SeasonID = @SeasonID AND TeamID = @HomeTeamID

print('One Update')

UPDATE TeamScore

SET Wins = Wins + 0, ConferenceWins = ConferenceWins + @AwayTeamConfWin,

GamesPlayed += 1, GamesWon += 0, DivisionWins = DivisionWins + @AwayTeamDivWin

WHERE

SeasonID = @SeasonID AND TeamID = @AwayTeamID

END

ELSE IF @HomeScore < @AwayScore

BEGIN

SELECT @Winner = 0

IF @HomeTeamDiv = @AwayTeamDiv

BEGIN

SELECT @HomeTeamDivWin = 0

SELECT @AwayTeamConfWin = 1

SELECT @HomeTeamConfWin = 0

SELECT @AwayTeamDivWin = 1

END

ELSE IF @HomeTeamConf = @AwayTeamConf

BEGIN

SELECT @HomeTeamDivWin = 0

SELECT @AwayTeamConfWin = 1

SELECT @HomeTeamConfWin = 0

SELECT @AwayTeamDivWin = 0

END

ELSE

SELECT @HomeTeamDivWin = 0

SELECT @AwayTeamConfWin = 0

SELECT @HomeTeamConfWin = 0

SELECT @AwayTeamDivWin = 0

UPDATE TeamScore

SET Wins = Wins + 0, ConferenceWins = ConferenceWins + @HomeTeamConfWin,

GamesPlayed += 1, GamesWon += 0, DivisionWins = DivisionWins + @HomeTeamDivWin

WHERE

SeasonID = @SeasonID AND TeamID = @HomeTeamID

UPDATE TeamScore

SET Wins = Wins + 1, ConferenceWins = ConferenceWins + @AwayTeamConfWin,

GamesPlayed += 1, GamesWon += 1, DivisionWins = DivisionWins + @AwayTeamDivWin

WHERE

SeasonID = @SeasonID AND TeamID = @AwayTeamID

END

UPDATE Game

SET HomeScore = @HomeScore, AwayScore = @AwayScore, Winner = @Winner

WHERE GameID = @GameID

PRINT('HOLA')

DECLARE @UserPick1 int;

SELECT @UserPick1 = UserPick FROM Picks WHERE GameID = @GameID AND UserID = 1

EXEC UpdateSeasonScore 1, @Season, @UserPick1, @Winner

DECLARE @UserPick2 int;

SELECT @UserPick2 = UserPick FROM Picks WHERE GameID = @GameID AND UserID = 2

EXEC UpdateSeasonScore 2, @Season, @UserPick2, @Winner

DECLARE @UserPick3 int;

SELECT @UserPick3 = UserPick FROM Picks WHERE GameID = @GameID AND UserID = 3

EXEC UpdateSeasonScore 3, @Season, @UserPick3, @Winner

DECLARE @UserPick4 int;

SELECT @UserPick4 = UserPick FROM Picks WHERE GameID = @GameID AND UserID = 4

EXEC UpdateSeasonScore 4, @Season, @UserPick4, @Winner

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

END CATCH

END

EXEC GameScore'Packers', 'Cowboys', 2019, 1, 41, 0

EXEC GameScore 'Chiefs', 'Patriots', 2019, 1, 0, 2

EXEC GameScore '49rs', 'Saints', 2019, 1, 27, 7

EXEC GameScore 'Titans', 'Steelers', 2019, 1, 7, 13

SELECT \* FROM Game

SELECT \* FROM TeamScore

SELECT \* FROM SeasonScore

A close up of a map

Description automatically generated

**Answering Data Questions:**

One of the most important parts of my solution includes displaying one weeks picks for everyone in my family. With this query, we can easily see this:

CREATE VIEW UserPick

AS

SELECT Picks.\*, Users.UserName, Users.FirstName, Users.LastName FROM Picks, Users WHERE Users.UserID = Picks.UserID

GO

CREATE VIEW GameView

AS

SELECT Game.\*, a.TeamName AS 'HomeTeam', b.TeamName AS 'AwayTeam' FROM Game, TeamInfo AS a, TeamInfo AS b

WHERE Game.AwayTeamID = b.TeamID AND Game.HomeTeamID = a.TeamID

GO

CREATE VIEW WeekOneView

AS

SELECT UserPick.UserName,

UserPick.UserPick,

GameView.HomeTeam,

GameView.AwayTeam,

GameView.HomeScore,

GameView.AwayScore,

GameView.WeekOfSeason,

GameView.Winner

FROM UserPick, GameView WHERE GameView.WeekOfSeason = 1 AND UserPick.GameID = GameView.GameID

GO

CREATE VIEW WeekOnePicks

AS

WITH Pivoted

AS

(

SELECT \* FROM WeekOneView

PIVOT

(

MAX([UserPick]) FOR [UserName] IN ( eramos,

hramos, mramos, pramos)

) AS p

)

SELECT \* FROM Pivoted

GO

SELECT \* FROM UserPick

SELECT \* FROM GameView

SELECT \* FROM WeekOneView

SELECT \* FROM WeekOnePicks

A screenshot of a cell phone

Description automatically generated

I created a couple of views in order to make it easier to display this. I can easily change the week and season in the view in order to get the week I wish to look at.

Another important requirement for the system is to compute every user’s score after a game’s score is put into the Game Table. The GameScore procedure helps us do this by taking a couple of inputs and updating three different tables: Game, SeasonScore, and TeamScore. It is also very simple to query in simple statistics like which team has won the most games:

WITH SeasonScore2019 AS (SELECT \* FROM TeamScore WHERE SeasonID = (SELECT SeasonID FROM Seasons WHERE Season=2019) )

SELECT \* FROM SeasonScore2019 WHERE Wins = (SELECT MAX(Wins) FROM SeasonScore2019)

A screenshot of a cell phone

Description automatically generated

Finally, we need to be able to track who has gotten the most correct predictions. This can be easily seen with this query:

SELECT SeasonScore.\*, Users.\* FROM SeasonScore, Users WHERE Users.UserID = SeasonScore.UserID

A screenshot of a video game

Description automatically generated

**Implementation:**

I decided to create my user interference using Microsoft Access. The only problem I have with Microsoft Access is that I do not like how I need to define the relationships. I prefer using Java NetBeans or Python in order to create the interface. I also found it easier to create a view in Microsoft SQL Server Management Studio and import the View instead of creating a query by defining the relationships in Access.

One of the main requirements is to be able to see who picked which team for one week. In MS Access I created a report using a View I created.

A screenshot of a computer

Description automatically generated

Then I also created a report to be able to see all the games in the season.

A screenshot of a computer

Description automatically generated

Finally, I created another report to see who is in the lead out of all in my family. (Of course, I am winning).

A screenshot of a social media post

Description automatically generated

**Reflection:**

At the beginning of the project, keeping track of who is gets a point after a game ends seemed impossible. While I do use SQL almost every day for work, I have never learned to manage it. It has only been SELECT statements for me, but I never realized how many other things happen in the background. The problem I see the most is the fact that some factory data tables are not normalized, so I spend most of my time as a Data Scientist cleaning up the data before I can analyze or model it.

I started the first part of the project after we talked about normalization and it is hard to create a conceptual model when you already know that it is not likely to give you the best results when you actually create the database. I stuck with only one to many relationships in order to save time when I had to normalize it. It is such a simple but crucial step that can save a significant amount of space in the database. The first thing I would do again is to create the best Normalized model that saves me the most space even if it complicates the queries by having to join more tables together.

One thing that did change from the first part of the project is adding the Seasons table. My model before this lacked the ability to differentiate the seasons by a simple join to the Seasons table. I considered making a Week and Seasons table as well, but most of the time you do not want to display the score for that week only, but the score for the whole season instead.

**Summary 2:**

From the first part to the second part not much changed except the inclusion of the Season table as I mentioned in the Reflection. The project creating was done very smoothly since the Normalized Form was very well fitting to the customer’s request. The one problem with the database is that a lot of the columns are calculated columns after a Game ends. Therefore, I had to create a procedure that took care of all the dependencies on the score of one game. This includes updating all the User’s score on the Season Score table and updating the Team Score Table for the two teams involved in the game.

In the end, once the dependencies were taken care of in the procedure, the Database is very user friendly. Since I am the main manager of the database, I can simply use the pre-maid procedures to add more Games in the Game table. It is also convenient how there is not going to be any more users included in the database, so I can create code that takes use of the four users only instead of a non-fixed number of users. Finally, thanks to Microsoft Access, I can easily share neat tables full of the information required for the solution in a very user-friendly manner.