Nihal Punjani
ID: 002000872
Final Project Report

Introduction:

The database I have decided to create is a Professional Basketball League Database. A professional basketball league has members which are Teams, Players, Coaches and Referees. There are a total of 30 team entities. Each team has multiple players and coaches, but players and coaches can only be hired by one team. Players and coaches are paid salary by the teams, and teams have salary caps determined by the league. The league hosts games at arenas at one of the participating team's arenas or an arena of choice. The referees monitor and enforce rules at the games and there must be a referee at each game. A special id record of all players, coaches, teams, and referees are kept in the leagues records department. Each game accumulates a large amount of statistical data for each team's performance or game statistics. The games are all a part of one season. The season has two different parts which are the regular season and postseason. The regular season is the mandatory games teams must participate in. A postseason is a playoff/tournament to compete for the championship. All teams participate in the regular season and the regular season determines who participates in the postseason. The postseason results in a champion. Also Awards are given at the end of the regular season. The database will be used to keep track of league history and statistical analysis of teams. There are 11 entities for this database. They are League Records, Team, Players, Coaches, Referees, Game, Game Stats, Arena, Season, Awards, and League Champion.

Data Requirements:

- League_Records: Keeps track of all ID numbers assigned to league members and type of entity they are
- Team: stores league's team's data.
- Player: stores the league's player's data
- Coach: stores the league's coach's data
- Referee: stores data of referees who work for the league.
- Game: stores the date and entity participation of games played.
- Performance: stores data of team performance and game's result
- Arena: stores data of the location where a game/games is played.
- Season: stores data of the date of season beginning and end
- Awards: stores data of award winners.
- League_Champion: stores data of the team that wins the postseason/finals.

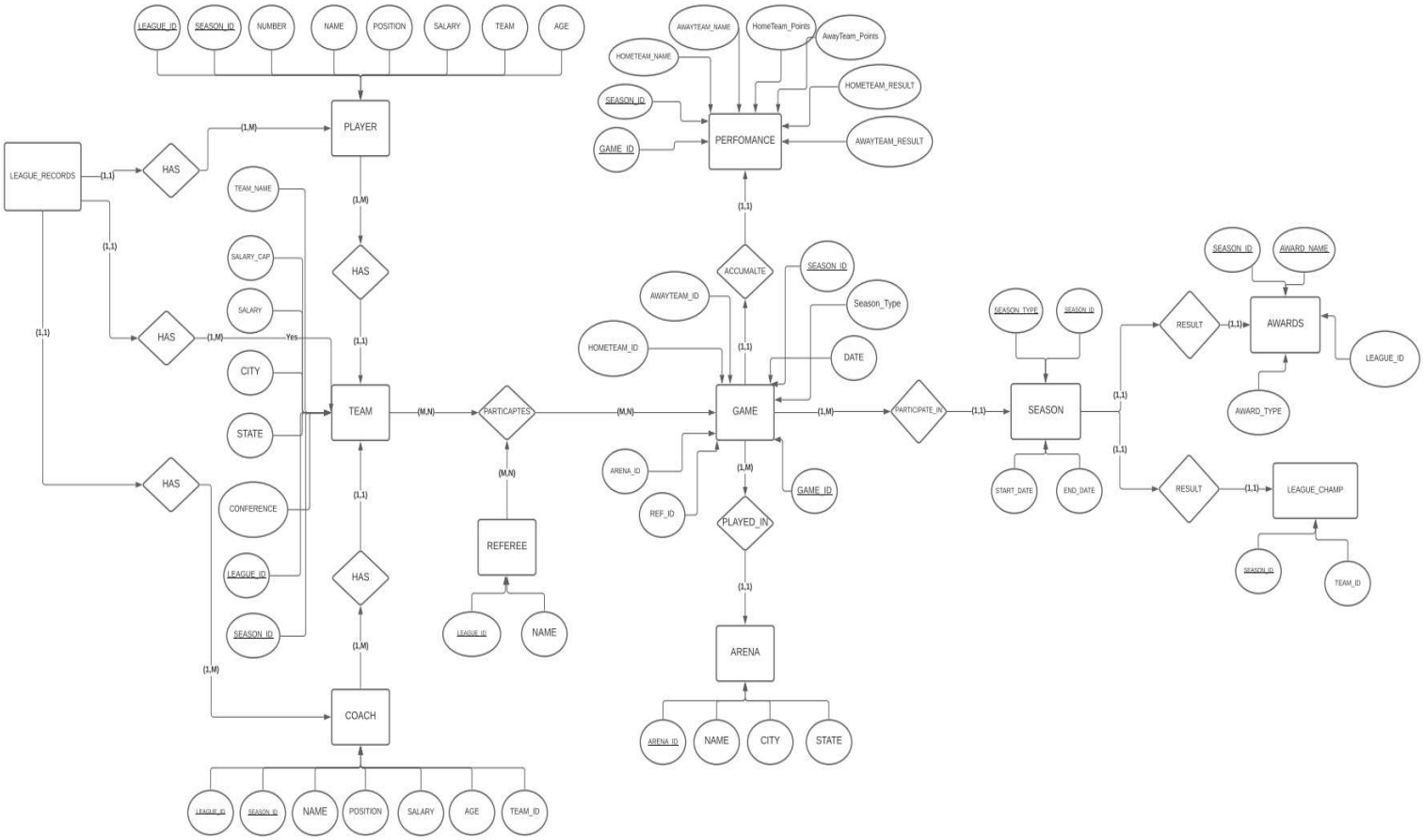
Functional Requirements:

- Find team data by season. Data such as season roster, coach, salary.

- Find Season total game record of all teams
- Find results of games by date range
- Find teams total wins and losses for the season against each team.
- Find schedule of games in date range or by team

This database is intended to be used openly by any user. Since the majority of professional sports data is open to the public, fans or analysts can access the database to see past results, upcoming games, or sports statistics.

EER Model:

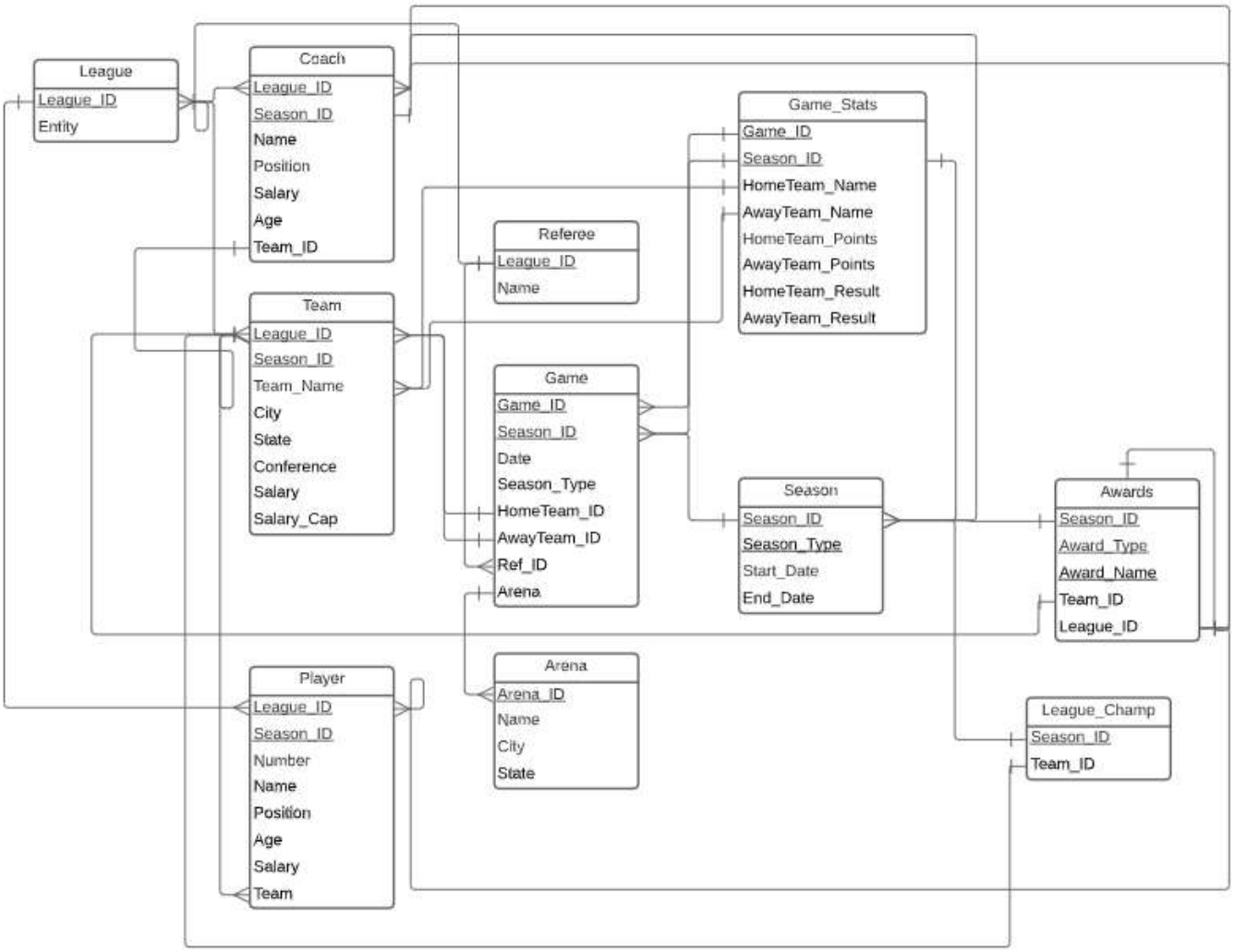


Primary and Foreign Keys:

- TEAM:
  - PK = (TEAM_ID, SEASON_ID)
  - FK = HEAD_COACH, SEASON_ID
- COACH
  - PK = (COACH_ID,SEASON_ID)
  - FK = TEAM, SEASON_ID

- PLAYER
    - PK = (PLAYER_ID, SEASON_ID)
    - FK = TEAM, SEASON_ID
- REFEREE
    - PK = REF_ID
- GAME
    - PK = (GAME_ID, SEASON_ID)
    - FK = HOME_TEAM, AWAY_TEAM, REF_ID, SEASON_ID, ARENA
- PERFORMANCE
    - PK = (GAME_ID, SEASON_ID, TEAM_ID)
    - FK = TEAM_ID, SEASON_ID
- ARENA
    - PK = ARENA_ID
- SEASON
    - PK = (SEASON_ID,SEASON_TYPE)
- LEAGUE_CHAMP
    - PK = SEASON_ID
    - FK = TEAM_ID

Relational Model:



The majority of cardinalities in this schema are 1:M or M:N. This affected my ability to assign a single attribute as a primary key for a lot of my entities. Since there can be so many changes among the entities, I chose to use composite keys with Season_ID, season year, and whichever attribute fit the entity. Records in sports databases like this must be organized by season. Even if something changes in the future it does not affect the past data. In order to find the correct instance of data the Season data is necessary.

Data Dictionary:

LEAGUE_RECORDS

Column	Data Type	Constraints	Keys	Description

LEAGUE_ID	INT	UNIQUE,NOT NULL	PRIMARY KEY	INTEGER
ENTITY	VARCHAR(20)			Characters up to 20

TEAM

Column	Data Type	Constraints	Keys	Description
LEAGUE_ID	INT	UNIQUE,NOT NULL	COMPOSITE PRIMARY KEY FOREIGN KEY->LEAGUE_RECORDS	INTEGER
Season_ID	VARCHAR(6)	UNIQUE,NOT NULL	COMPOSITE PRIMARY KEY	1- 5 characters. Must start with either R or P and end with 4 digits for year
Team_Name	VARCHAR(30)	UNIQUE, NOT NULL		1- 20 characters only letters
City	VARCHAR(20)	NOT NULL		1- 20 characters only letters
State	VARCHAR(2)	NOT NULL		1- 2 characters only letters, state abbrv.
Conference	VARCHAR(7)	NOT NULL Conference = EASTERN or Conference = WESTERN		1- 7 characters only letters
Salary	INT	NOT NULL Salary < Salary_Cap		10 digit integer
Salary_Cap	INT	NOT NULL Salary_Cap > Salary		10 digit integer

Player

Column	Data Type	Constraints	Keys	Description
League_ID	INT	UNIQUE,NOT NULL	COMPOSITE PRIMARY KEY	4 digit integer
Season_ID	VARCHAR(6)	NOT NULL	COMPOSITE PRIMARY KEY	1- 5 characters. Must start with either R or P and end with 4 digits for year
Team_Number	VARCHAR(2)	NOT NULL		2 characters that are digit since some players have 00 number need varchar
Full_Name	VARCHAR(20)	NOT NULL		1- 20 characters only letters
Position	VARCHAR(20)	NOT NULL		1- 20 characters only letters, state abbrv.
Age	INT(2)	NOT NULL Age > 17		2 digit integer
Salary	INT(10)	NOT NULL Salary < Salary_Cap		10 digit integer
Team_ID	INT	NOT NULL	FOREIGN KEY -> TEAM(LEAGUE_ID)	2 digit integer

Coach

Column	Data Type	Constraints	Keys	Description
League_ID	INT	UNIQUE, NOT NULL	COMPOSITE PRIMARY KEY FOREIGN KEY ->	3 digit integer

			LEAGUE_RECORDS	
Season_ID	VARCHAR(6)	NOT NULL	COMPOSITE PRIMARY KEY	1- 5 characters. Must start with either R or P and end with 4 digits for year
Name	VARCHAR(20)	NOT NULL		1- 20 characters only letters
Position	VARCHAR(20)	NOT NULL		1- 20 characters
Salary	INT (10)	NOT NULL SALARY < SALARY_CAP		10 digit integer
Age	INT (2)	NOT NULL		2 digit integer
Team_ID	INT	NOT NULL	FOREIGN KEY->TEAM	Integer variable

Referee

Column	Data Type	Constraints	Keys	Description
League_ID	INT	UNIQUE, NOT NULL	PRIMARY KEY FOREIGN KEY -> League_Records	3 digit integer
Name	VARCHAR(20)	NOT NULL		1- 20 characters only letters

Season

Column	Data Type	Constraints	Keys	Description
Season_ID	VARCHAR(6)	UNIQUE, NOT NULL	COMPOSITE PRIMARY KEY	1- 5 characters. Must start with either R or P and end with 4 digits for year
Season_Type	VARCHAR(10)	UNIQUE, NOT	COMPOSITE	1-10 Characters,

		NULL	PRIMARY KEY	letters only
Start_Date	INT(4)	NOT NULL START_DATE <= END_DATE		4 digit integers, month is first 2 digits and day is last 2.
End_Date	INT(4)	NOT NULL START_DATE <= END_DATE		4 digit integers, month is first 2 digits and day is last 2.

Game

Column	Data Type	Constraints	Keys	Description
Game_ID	INT (1023)	UNIQUE	COMPOSITE PRIMARY KEY	1
Season_ID	VARCHAR(6)	UNIQUE, NOT NULL	COMPOSITE PRIMARY KEY COMPOSITE FOREIGN KEY->SEASON	1- 5 characters. Must start with either R or P and end with 4 digits for year
Season_Type	VARCHAR(20)	NOT NULL	COMPOSITE FOREIGN KEY->SEASON	1- 20 characters, letters only
Date	INT(4)	NOT NULL		4 digit integers, month is first 2 digits and day is last 2.
Home_Team	INT(2)	NOT NULL	FOREIGN KEY->TEAM	2 digit integer
Away_Team	INT(2)	NOT NULL	FOREIGN KEY->TEAM	2 digit integer
Ref_ID	INT(3)	UNIQUE, NOT NULL	FOREIGN KEY->REFEREE	3 digit integer
Arena_ID	INT(3)	NOT NULL	FOREIGN KEY->ARENA	3 digit integer

Arena

Column	Data Type	Constraints	Keys	Description
Arena_ID	INT(3)	UNIQUE, NOT NULL	PRIMARY KEY	3 digit integer
Name	VARCHAR(50)	NOT NULL		1- 50 characters
City	Varchar(20)	NOT NULL		1- 20 characters
State	Varchar(2)	NOT NULL		1- 2 characters only letters

GAME_STATS

Column	Data Type	Constraints	Keys	Description
Game_ID	INT (3)	UNIQUE, NOT NULL	COMPOSITE PRIMARY KEY COMPOSITE FOREIGN KEY->GAME COMPOSITE FOREIGN KEY->SEASON	3 digit integer
Season_ID	VARCHAR(6)	NOT NULL	COMPOSITE PRIMARY KEY COMPOSITE FOREIGN KEY->GAME COMPOSITE FOREIGN KEY->SEASON	1- 5 characters. Must start with either R or P and end with 4 digits for year
HomeTeam_Name	VarChar(30)	UNIQUE, NOT NULL		Up to 30 characters only letters
AwayTeam_Name	VarChar(30)	NOT NULL		Up to 30 characters only

				letters
HomeTeam_Points	INT	NOT NULL		Up to 3 digit integer variable
AwayTeam_Points	INT	NOT NULL		Up to 3 digit integer variable
HomeTeam_Result	VARCHAR(5)	NOT NULL		Up to 5 characters only letters
AwayTeam_Result	VARCHAR(5)	NOT NULL		Up to 5 characters only letters

Awards

Column	Data Type	Constraints	Keys	Description
Season_ID	VARCHAR(6)	UNIQUE, NOT NULL	COMPOSITE PRIMARY KEY	1- 5 characters. Must start with either R or P and end with 4 digits for year
Award_Type	VARCHAR(10)	UNIQUE, NOT NULL	COMPOSITE PRIMARY KEY	1-10 Characters, letters only
MVP	INT(3)	NOT NULL	FOREIGN KEY->PLAYER	2 digit integer
Coach_Of_The_Year	INT(3)	NOT NULL	FOREIGN KEY->COACH	2 digit integer

League_Champ

Column	Data Type	Constraints	Keys	Description
Season_ID	VARCHAR(6)	UNIQUE, NOT NULL	COMPOSITE PRIMARY KEY	1- 5 characters. Must start with either R or P and end with 4 digits

				for year
Team_ID	INT(2)	UNIQUE,NOT NULL	FOREIGN KEY-> TEAM	2 digit integer

IMPLEMENTATION

Source Code:

```
CREATE SCHEMA BB_LEAGUE;

USE BB_LEAGUE;


DROP TABLE IF EXISTS LEAGUE_RECORDS;

#CREATE TABLE STATEMENT

CREATE TABLE LEAGUE_RECORDS (

        LEAGUE_ID INT NOT NULL PRIMARY KEY,

   ENTITY VARCHAR(20)

);


DROP TABLE IF EXISTS TEAM;

#CREATE TABLE STATEMENT

CREATE TABLE TEAM (

        LEAGUE_ID INT NOT NULL,

   SEASON_ID VARCHAR (6) NOT NULL,

   TEAM_NAME VARCHAR (30) NOT NULL,

   CITY VARCHAR (20) NOT NULL,

   STATE VARCHAR (2) NOT NULL,
```

```
    CONFERENCE VARCHAR (7),
    SALARY INT NOT NULL,
    SALARY_CAP INT NOT NULL,
    PRIMARY KEY (LEAGUE_ID,SEASON_ID),
    FOREIGN KEY (LEAGUE_ID) REFERENCES LEAGUE_RECORDS(LEAGUE_ID) ON
DELETE CASCADE ON UPDATE CASCADE
);

DROP TABLE IF EXISTS PLAYER;
#CREATE TABLE STATEMENT
CREATE TABLE PLAYER (
        LEAGUE_ID INT NOT NULL,
    SEASON_ID VARCHAR (6) NOT NULL,
    TEAM_NUMBER INT NOT NULL,
    FULL_NAME VARCHAR(20) NOT NULL,
    POSITION VARCHAR(20) NOT NULL,
    AGE TINYINT NOT NULL,
    SALARY BIGINT NOT NULL,
    TEAM INT NOT NULL,
    PRIMARY KEY (LEAGUE_ID,SEASON_ID),
    UNIQUE(LEAGUE_ID),
    FOREIGN KEY (LEAGUE_ID) REFERENCES LEAGUE_RECORDS(LEAGUE_ID) ON
DELETE CASCADE ON UPDATE CASCADE
```

```
);

DROP TABLE IF EXISTS COACH;
#CREATE TABLE STATEMENT
CREATE TABLE COACH (
        LEAGUE_ID INT NOT NULL,
    SEASON_ID VARCHAR(6) NOT NULL,
    FULL_NAME VARCHAR(20) NOT NULL,
    POSITION VARCHAR(20) NOT NULL,
    AGE INT NOT NULL,
    SALARY INT NOT NULL,
    TEAM_ID INT NOT NULL,
    PRIMARY KEY (LEAGUE_ID,SEASON_ID),
    UNIQUE(LEAGUE_ID),
    FOREIGN KEY (LEAGUE_ID) REFERENCES LEAGUE_RECORDS(LEAGUE_ID) ON
DELETE CASCADE ON UPDATE CASCADE
);

DROP TABLE IF EXISTS REFEREE;
#CREATE TABLE STATEMENT
CREATE TABLE REFEREE (
        LEAGUE_ID INT NOT NULL PRIMARY KEY,
    FULL_NAME VARCHAR(20) NOT NULL,
```

```
    AGE INT NOT NULL,
    FOREIGN KEY (LEAGUE_ID) REFERENCES LEAGUE_RECORDS(LEAGUE_ID) ON
DELETE CASCADE ON UPDATE CASCADE
);

DROP TABLE IF EXISTS ARENA;
#CREATE TABLE STATEMENT
CREATE TABLE ARENA(
        ARENA_ID TINYINT PRIMARY KEY,
    ARENA_NAME VARCHAR(30) NOT NULL,
    CITY VARCHAR(20) NOT NULL,
    STATE VARCHAR(2) NOT NULL
);

DROP TABLE IF EXISTS SEASON;
#CREATE TABLE STATEMENT
CREATE TABLE SEASON(
        SEASON_ID VARCHAR(6) NOT NULL,
    SEASON_TYPE VARCHAR(10) NOT NULL,
    START_DATE DATE NOT NULL,
    END_DATE DATE NOT NULL,
    PRIMARY KEY(SEASON_ID,SEASON_TYPE)
);
```

```
DROP TABLE IF EXISTS GAME;
#CREATE TABLE STATEMENT
CREATE TABLE GAME (
        GAME_ID INT NOT NULL,
    SEASON_ID VARCHAR (6) NOT NULL,
    GAME_DATE DATE NOT NULL,
    SEASON_TYPE VARCHAR(20) NOT NULL,
    HOMETEAM_ID INT NOT NULL,
    AWAYTEAM_ID INT NOT NULL,
    REF_ID INT NOT NULL,
    ARENA_ID TINYINT NOT NULL,
    PRIMARY KEY(GAME_ID, SEASON_ID),
    FOREIGN KEY (HOMETEAM_ID) REFERENCES TEAM(LEAGUE_ID) ON DELETE
CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (AWAYTEAM_ID) REFERENCES TEAM(LEAGUE_ID) ON DELETE
CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (REF_ID) REFERENCES REFEREE(LEAGUE_ID) ON DELETE CASCADE
ON UPDATE CASCADE,
    FOREIGN KEY (ARENA_ID) REFERENCES ARENA(ARENA_ID) ON DELETE CASCADE
ON UPDATE CASCADE,
        FOREIGN KEY (SEASON_ID,SEASON_TYPE) REFERENCES
SEASON(SEASON_ID,SEASON_TYPE) ON DELETE CASCADE ON UPDATE CASCADE
```

```
);


DROP TABLE IF EXISTS GAME_STATS;
#CREATE TABLE STATEMENT
CREATE TABLE GAME_STATS(
        GAME_ID INT NOT NULL,
   SEASON_ID VARCHAR (6) NOT NULL,
   HOMETEAM_NAME VARCHAR(30) NOT NULL,
   AWAYTEAM_NAME VARCHAR(30) NOT NULL,
   HOMETEAM_POINTS INT NOT NULL,
   AWAYTEAM_POINTS INT NOT NULL,
   HOMETEAM_RESULT VARCHAR(5) NOT NULL,
   AWAYTEAM_RESULT VARCHAR(5) NOT NULL,
   PRIMARY KEY(GAME_ID,SEASON_ID),
   FOREIGN KEY (GAME_ID,SEASON_ID) REFERENCES GAME(GAME_ID,SEASON_ID) ON
DELETE CASCADE ON UPDATE CASCADE,
   FOREIGN KEY (GAME_ID,SEASON_ID) REFERENCES GAME(GAME_ID,SEASON_ID) ON
DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY (GAME_ID,SEASON_ID) REFERENCES
GAME(GAME_ID,SEASON_ID) ON DELETE CASCADE ON UPDATE CASCADE,
   FOREIGN KEY (GAME_ID,SEASON_ID) REFERENCES GAME(GAME_ID,SEASON_ID) ON
DELETE CASCADE ON UPDATE CASCADE
);
```

```
DROP TABLE IF EXISTS LEAGUE_CHAMPS;
#CREATE TABLE STATEMENT
CREATE TABLE LEAGUE_CHAMPS(
        SEASON_ID INT NOT NULL PRIMARY KEY,
   TEAM_ID INT NOT NULL,
   FOREIGN KEY(TEAM_ID) REFERENCES LEAGUE_RECORDS(LEAGUE_ID) ON DELETE
CASCADE ON UPDATE CASCADE
);

DROP TABLE IF EXISTS AWARDS;
#CREATE TABLE STATEMENT
CREATE TABLE AWARDS(
        SEASON_ID VARCHAR(6) NOT NULL,
   AWARD_NAME VARCHAR(30) NOT NULL,
   AWARD_TYPE VARCHAR(10) NOT NULL,
   LEAGUE_ID INT NOT NULL,
   FOREIGN KEY (LEAGUE_ID) REFERENCES LEAGUE_RECORDS(LEAGUE_ID) ON
DELETE CASCADE ON UPDATE CASCADE
);
```

#queries

```
# See current roster and coach with salaries
#Basic Query to test data base
```

```
SELECT HOMETEAM_NAME AS TEAM, ROUND(AVG(HOMETEAM_POINTS),2) as
AVG_POINTS_HOME, ROUND(AVG(AWAYTEAM_POINTS),2) as AVG_POINTS_AWAY
FROM GAME_STATS JOIN TEAM ON GAME_STATS.HOMETEAM_NAME =
TEAM.TEAM_NAME
GROUP BY HOMETEAM_NAME
ORDER BY TEAM ASC;
```

```
#FD query to view season results per team
SELECT A.T_NAME AS TEAM_NAME, SUM(WIN) AS W, SUM(LOSS) AS L
FROM (SELECT  HOMETEAM_NAME AS T_NAME , SUM(HOMETEAM_RESULT = 'Win') AS
WIN, SUM(HOMETEAM_RESULT = 'Lose') AS LOSS
FROM GAME_STATS
GROUP BY HOMETEAM_NAME
UNION ALL
SELECT AWAYTEAM_NAME, SUM(AWAYTEAM_RESULT = 'Win') AS WIN,
SUM(AWAYTEAM_RESULT = 'Lose') AS LOSS
FROM GAME_STATS
GROUP BY AWAYTEAM_NAME) AS A
GROUP BY T_NAME
ORDER BY T_NAME;
```

#FD to see Season results of one Team against other team, Query will sum result of wins and losses against each opponent

```
SELECT A.OPPONENT AS ATLANA_VS_OPPONENT, SUM(WIN) AS W, SUM(LOSS) AS L
FROM (SELECT AWAYTEAM_NAME AS OPPONENT, SUM(HOMETEAM_RESULT = 'Win') AS
WIN, SUM(HOMETEAM_RESULT = 'Lose') AS LOSS
FROM GAME_STATS
WHERE HOMETEAM_NAME = "Atlanta Socks"
GROUP BY AWAYTEAM_NAME
UNION ALL
SELECT HOMETEAM_NAME, SUM(AWAYTEAM_RESULT = 'Win') AS WIN,
SUM(AWAYTEAM_RESULT = 'Lose') AS LOSS
FROM GAME_STATS
WHERE AWAYTEAM_NAME = "Atlanta Socks"
GROUP BY HOMETEAM_NAME) AS A
GROUP BY OPPONENT
ORDER BY OPPONENT ASC;
```

#FD query to see result of games in date range

```
SELECT GAME_DATE, HOMETEAM_NAME, HOMETEAM_POINTS, AWAYTEAM_NAME,
AWAYTEAM_POINTS
FROM GAME_STATS JOIN GAME ON GAME_STATS.GAME_ID = GAME.GAME_ID AND
GAME_STATS.SEASON_ID = GAME.SEASON_ID
```

```
WHERE GAME_DATE BETWEEN '2020-02-31' AND '2020-04-01'
ORDER BY GAME_DATE ASC;

#FD query to see salary of players and coaches of a team
SELECT TEAM_NAME, FULL_NAME,POSITION,P.SALARY, AGE
FROM (SELECT TEAM_ID,FULL_NAME,POSITION,SALARY, AGE
FROM PLAYER
WHERE TEAM_ID = 9
UNION ALL
SELECT TEAM_ID,FULL_NAME,POSITION, SALARY AS SALARY,AGE
FROM COACH
WHERE TEAM_ID = 9
ORDER BY SALARY) as P JOIN TEAM ON P.TEAM_ID = TEAM.LEAGUE_ID;
```

Views

```
DROP VIEW IF EXISTS LEAGUE_SALARIES;
#FD VIEW of SALARIES of every Player and Coach in League
CREATE VIEW LEAGUE_SALARIES  AS SELECT
TEAM_NAME,FULL_NAME,POSITION,SALARY,AGE
FROM (SELECT TEAM_NAME, FULL_NAME,POSITION,P.SALARY, AGE
FROM (SELECT TEAM_ID,FULL_NAME,POSITION,SALARY, AGE
FROM PLAYER
UNION ALL
SELECT TEAM_ID,FULL_NAME,POSITION, SALARY AS SALARY,AGE
```

```
FROM COACH
ORDER BY SALARY) as P JOIN TEAM ON P.TEAM_ID = TEAM.LEAGUE_ID) AS S;

#QUERY FOR ABOVE VIEW
SELECT *
FROM LEAGUE_SALARIES;

#FD CREATE VIEW OF SEASON GAME RESULTS
CREATE VIEW LEAGUE_RESULTS AS SELECT TEAM_NAME, W, L
FROM (SELECT A.T_NAME AS TEAM_NAME, SUM(WIN) AS W, SUM(LOSS) AS L
FROM (SELECT  HOMETEAM_NAME AS T_NAME , SUM(HOMETEAM_RESULT = 'Win') AS
WIN, SUM(HOMETEAM_RESULT = 'Lose') AS LOSS
FROM GAME_STATS
GROUP BY HOMETEAM_NAME
UNION ALL
SELECT AWAYTEAM_NAME, SUM(AWAYTEAM_RESULT = 'Win') AS WIN,
SUM(AWAYTEAM_RESULT = 'Lose') AS LOSS
FROM GAME_STATS
GROUP BY AWAYTEAM_NAME) AS A
GROUP BY T_NAME
ORDER BY T_NAME) AS B;

#QUERY FOR ABOVE VIEW
```

```
SELECT *
FROM LEAGUE_RESULTS;
```

Summary

The project has capabilities of giving past data and upcoming events. Basic search data of teams, coaches, seasons, games are available. The database is most useful with the ability to retrieve aggregate data. If a user wanted to know aggregate data at specific events such as which team do they have the most wins against or how many away games does a team win is capable through the database.