**Table of Contents**

[**1. Introduction**](#_8mg32xgxrjvx) **3**

[**2. Application**](#_xnydfwjggxjq) **3**

[2.1 Application of database](#_xdxh0uvg98pj) 3

[2.2 Data Held in Database](#_vlc24voj4t5g) 3

[2.3 Use Case Queries](#_c1d3ls89hojv) 3

[**3. Entities**](#_pvmiictf0luk) **4**

[**4. Design**](#_ce05twfef5wz) **5**

[4.1 Assumptions](#_tzfgl8db85v6) 5

[**4.2 Entity-Relationship Diagram**](#_7y9thlnoqdjg) **6**

[Figure 1: Entity-Relationship Diagram of NFL Database](#_9xc89ux7hs87) 6

[**4.3 Relational Data Model**](#_g7nm3k99joiq) **7**

[Figure 2: Relational Data Model of NFL Database](#_lzhnia955uib) 7

[4.4 Constraints](#_knt5w9scobw7) 8

[PLAYER](#_a9c317a6e8z6) 8

[TEAM](#_5x2ibmmz8tl6) 8

[COACH](#_jzhf6ya357ip) 8

[GAME](#_i16ygcc2da4y) 8

[STADIUM](#_pewt4omglngr) 8

[OFFENSIVE RANKING](#_q4fip5s30h2p) 8

[DEFENSIVE RANKING](#_bhvk01bgpc8r) 9

[PARTICIPATES\_IN](#_ysbwwnnu1dmq) 9

[**5. Schedule**](#_holhtwjhtpgo) **10**

[Table 1: Approximate Schedule for Deliverables](#_ekkafvtp2jn4) 10

[**6. Tooling**](#_evrzfvcszkn8) **11**

[6.1 Database Management System](#_47ty4qr3bbi8) 11

[6.2 User Interface](#_pbhdbj463xga) 11

[6.3 Database Hosting](#_miu3tk1vwdth) 11

[6.4 Support Tooling](#_z3c9i8bahajm) 11

[6.5 Data for Database](#_pgf3moys39tp) 11

[**7. Tables Creation**](#_eofrq1dmpa99) **12**

[**8. Table Inserts**](#_rskzv67rpfzx) **14**

[Table TEAM inserts](#_525ya75kfe4m) 14

[Table STADIUM inserts](#_e4padbjbcx1k) 14

[Table PLAYER inserts](#_snqsiral4v9w) 14

[Table COACH inserts](#_9vhjt4ej9emu) 14

[Table OFFENSIVE\_RANKING inserts](#_xl52g5hofrj1) 14

[Table DEFENSIVE\_RANKING inserts](#_w4wtt0bee9ul) 14

[Table GAME inserts](#_eracn2hd1czn) 14

[Table PARTICIPATES\_IN inserts](#_o1lp3hdkvu56) 15

[**9. Creation and Processing of Test Data**](#_z0s9o4stz05k) **16**

[**10. Documented Changes**](#_7z60thzd13rn) **17**

[10.1 Reasoning for Changes](#_3ybqu0o7shfo) 17

[10.2 List of Changes](#_pxlhinv6ytd1) 17

[**11. Queries**](#_6wz9i93ylgkl) **19**

[Table 2: List of Query Statements and their Purpose](#_t7v2v7j93hxr) 19

[**12. Normalization**](#_2tiq66ne708o) **22**

[Table 3: Functional Dependencies and Normal Forms](#_8q8xgz6tb9gn) 22

[**13. Discussion**](#_eum1fjk098hl) **25**

[13.1 Evaluation of the NFL Database Project](#_qiyyhc6y9s7m) 25

[13.2 Feedback from Document Iteration Submissions](#_i6e58ukc537n) 25

[13.3 Generation of Data, Population of database, and Testing](#_w0z8by49eapj) 26

[**14. Citations**](#_l73rqw92z00t) **27**

# 

# **1. Introduction**

DataDawgs intends to create a National Football League database that will help store and monitor data on the thirty-two football teams in the NFL and the games played during the season.

# **2. Application**

## **2.1 Application of database**

The application will be utilized by the National Football League to store and monitor data on all thirty-two football teams that are affiliated with the National Football League from 2018 season, which include data on each of the teams, players, coaches, games played, and information of the stadiums where the games take place.

## **2.2 Data Held in Database**

The data held in the database will include player details and statistics (position, salary, name, games played), team details (wins, losses, ranking) and statistics, game schedules (dates, times, locations), scoring information, coach information (name, position, salary), stadium information (name, location, capacity), play statistics (longest throw, longest run, longest field goal), game information and etc …

## **2.3 Use Case Queries**

**Retrieve** all player\_names whose team\_name = “Seahawks”;

**Delete** player\_id = 666346 from team\_name = “Seahawks”;

**Update** stadium\_ID = 19 stadium\_name to “Century Link”;

**Insert** a new PLAYER in the database whose player\_id = 178920, team\_name = “Seahawks”, player\_name = “Wilson Russell”, position = “QB”;

**Insert** a new STADIUM in the database whose stadium\_ID = “23”, stadium\_name = “Gillette Stadium”, location\_city= “Foxborough”, location\_state = “MA”;

# 3. Entities

The following are the entity data types that are in the NFL database:

* **PLAYER:** player\_ID , team\_name, player\_name, position
* **TEAM:** team\_name, state\_represented
* **COACH:** coach\_ID, team\_name, coach\_fname, coach\_lname
* **GAME:** game\_ID, stadium\_ID, away\_team\_name, home\_team\_name, final\_score
* **STADIUM:** stadium\_ID, stadium\_name, location\_state, location\_city
* **OFFENSIVE\_RANKING:** team\_name , Ototal\_score, Ototal\_yards, Oturnovers, Openalties
* **DEFENSIVE\_RANKING:** team\_name , Dtotal\_score, Dtotal\_yards, Dturnovers, Dpenalties

# 

# **4. Design**

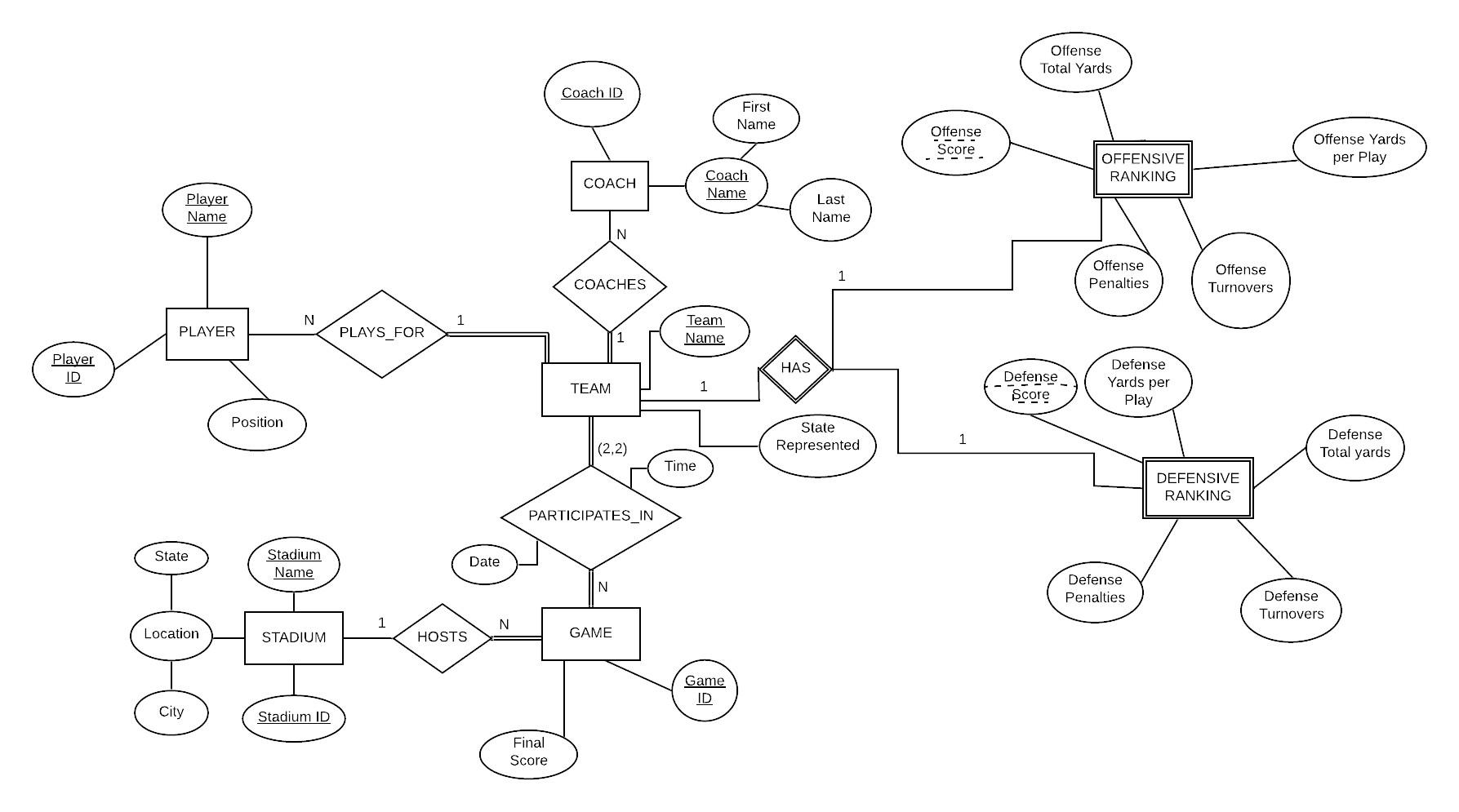
## 4.1 Assumptions

1. A COACH has a unique coach\_ID.
2. A PLAYER has a unique player\_ID.
3. A GAME has a unique game\_ID.
4. Two TEAMs participate in each GAME.
5. A STADIUM has a unique stadium\_name and a unique stadium\_ID.
6. A STADIUM has a unique city it is located in.
7. Two TEAMs PARTICIPATE\_IN one game at a time (home and away team)
8. A TEAM can only PARTICIPATE\_IN one game on a specified date.
9. A TEAM must have a unique team\_name
10. The domain (yards) constraints for offensive attributes will also apply to defensive attributes as either a subset or a proper subset of the original set

## 4.2 Entity-Relationship Diagram

The diagram (Figure 1) below is a high-level conceptual data model that graphically represents entities and their relationships to one another and is used to map the external level to the conceptual level in the ANSI-SPARK three-level Architecture model. We are utilized this diagram to show the entities and relationships in the NFL database we are creating. These entities are; TEAM, PLAYER, COACH, STADIUM, GAME, OFFENSIVE RANKING, and DEFENSIVE RANKING.

### Figure 1: Entity-Relationship Diagram of NFL Database

****

## 4.3 Relational Data Model

The Relational Model (Figure 2) provides a more detailed outline of the relations between the entities and allows us to better organize the elements of data and see how they relate to each other. The relational data model corresponds to the mapping from the conceptual to internal level of the ANSI-SPARK model. In the process of mapping from ER to RM we chose our primary keys from a list of possible candidate keys. On the diagram keys are underlined and foreign keys are drawn. The color coding represents different foreign keys and there are cascading relationships to show the relationships between the entities’ keys.

### Figure 2: Relational Data Model of NFL Database

****

## 4.4 Constraints

### **PLAYER**

player\_ID (PK): unique int NOT NULL

team\_name (FK): alphabetical string of max length 50 NOT NULL

player\_name: alphabetical string of max length 50 NOT NULL format: last name first name

position: alphabetical string of max length 5 {“C”, “CB”, “DB”, “DE”, “DE/LB”, “DL”, “DT”, “FB”, “FS”, “G”, “ILB”, “K”, “KR”, “LB”, “LS”, “NT”, “OL”, “OLB”, “OT”, “P”, “QB”, “RB”, “S”, “SS”, “T”, “TE”, “WR”, “SAF”} NULL

### **TEAM**

team\_name (PK): alphabetical string of max length 50 NOT NULL

state\_represented: alphabetical string of max length 50 NULL

### **COACH**

coach\_ID (PK): unique int NOT NULL

team\_name (FK): alphabetical string of max length 50 NOT NULL

coach\_fname: alphabetical string of max length 50 NULL

coach\_lname: alphabetical string of max length 50 NULL

### **GAME**

game\_ID (PK): unique int NOT NULL

stadium\_ID (FK): unique int NOT NULL

home\_team (FK): alphabetical string of max length 30 NOT NULL,

away\_team (FK): alphabetical string of max length 30 NOT NULL,

final\_score: int NULL

### **STADIUM**

stadium\_ID (PK): unique int NOT NULL

stadium\_name: alphabetical string of max length 50 NULL

location\_state: alphabetical string of max length 50 NULL

location\_city: alphabetical string of max length 50 NULL

### **OFFENSIVE RANKING**

team\_name (FK): alphabetical string of max length 50 NOT NULL

Ototal\_score (PK): double {0.00, 100.00} NULL

Oyards\_per\_play: double {0.00, 10.00} NULL

Ototal\_yards: smallint NULL

Oturnovers: double {0.00, 10.00} NULL

Openalties: double {0.00, 20.00} NULL

### **DEFENSIVE RANKING**

team\_name (FK): alphabetical string of max length 50 NOT NULL

Dtotal\_score (PK): double {0.00, 100.00} NOT NULL

Dtotal\_yards: smallint NULL

Dturnovers: double {0.00, 10.00} NULL

Dpenalties: double {0.00, 20.00} NULL

### **PARTICIPATES\_IN**

home\_team\_name (PK): alphabetical string of max length 50 NOT NULL

away\_team\_name (PK): alphabetical string of max length 50 NOT NULL

game\_ID (PK): unique int NOT NULL

game\_date: numerical string of length 10 format: xxxx-xx-xx (year-month-day) NULL

# 

# **5. Schedule**

### Table 1: Approximate Schedule for Deliverables

|  |  |  |  |
| --- | --- | --- | --- |
| **Iteration** | **Task:** | **Completed By:** | **Team Member(s):** |
| 0 | Set Team By-Laws | 10/08/2019 | All |
| 0 | Write Project Proposal | 10/08/2019 | All |
| 1 | Create Entity-Relationship Diagram | 10/17/2019 | All |
| 1 | Create Relational Schema Model | 10/17/2019 | All |
| 2 | Choose Database Software | 10/24/2019 | All |
| 2 | Choose Database Hosting | 10/24/2019 | All |
| 2 | Create Tooling PowerPoint | 10/27/2019 | All |
| 2 | Tooling Presentation | 10/30/2019 | All |
| 4 | Normalization | 12/01/2019 | Alex Pantile |
| 3 | Create Database | 11/17/2019 | Eyas Rashid |
| 3 | Populate Database | 11/24/2019 | All |
| 3 | Write SQL Query Statements | 11/24/2019 | Vineet Ramisetty & Alex Pantile |
| 4 | Create User Interface | 12/01/2019 | Vineet Ramisetty |
| 4 | Create Poster | 12/08/2019 | Hadassah Latchague |
| 4 | Documentation and Poster Presentation | 12/11/2019 | All |

# **6. Tooling**

## **6.1 Database Management System**

We have decided on using MySQL and MySQL Workbench for our database management system because it is a popular tool. In addition, we can find support for how to use this tool because of its popularity and because it is open source.

## **6.2 User Interface**

For our user interface, we have decided on a console application that will be implemented using Java. All of our group has experience using Java to build UI and front-end interfaces. In addition, we will be able to find help and resources online due to Java’s popularity.

## **6.3 Database Hosting**

We have decided to use Amazon Web Services to host our database because it’s easy to use, secure, and flexible. We have a team member whose relative has a lot of experience hosting on AWS, therefore this will allow us to receive help and ask questions if needed. AWS also has a lot of tutorials to help guide users on how to use MySQL in AWS.

## **6.4 Support Tooling**

We are planning on using GitHub for version control as all group members have experience using github. With github we can communicate remotely and continuously check on an improve our code. Github and Git are also popular tools so online resources and help will be easy to find.

## **6.5 Data for Databas**e

We will gather our data directly from the NFL:<https://www.nfl.com/> The NFL keeps track of all the data that we need for our project and also keeps track of a lot more. As a team we have narrowed down what data is necessary for our database and we plan to use this data only. The data is all real-world data from the 2018 NFL season. We limited the timeframe to only one year’s data to reduce complexity.

# 

# **7. Tables Creation**

This section contains table creations of each table in the NFL database.

CREATE TABLE TEAM (

team\_name VARCHAR(30) NOT NULL UNIQUE,

state\_represented VARCHAR(50) NULL,

PRIMARY KEY (team\_name)

);

CREATE TABLE STADIUM (

stadium\_ID INT NOT NULL UNIQUE,

stadium\_name VARCHAR(30) NOT NULL UNIQUE,

location\_state VARCHAR(15),

location\_city VARCHAR(15),

PRIMARY KEY(stadium\_ID)

);

CREATE TABLE PLAYER (

player\_ID INT NOT NULL UNIQUE,

team\_name VARCHAR(30) NOT NULL,

player\_name VARCHAR(50) NOT NULL,

position VARCHAR(5),

PRIMARY KEY(player\_ID),

FOREIGN KEY(team\_name) REFERENCES TEAM(team\_name),

CHECK (position IN('C', 'CB', 'DB', 'DE', 'DE/LB', 'DL',

'DT', 'FB', 'FS', 'G', 'ILB', 'K', 'KR', 'LB', 'LS', 'NT',

'OL', 'OLB', 'OT', 'P', 'QB', 'RB', 'S', 'SS', 'T', 'TE', 'WR', 'LB', 'OG', 'MLB', 'SAF'))

);

CREATE TABLE COACH (

team\_name VARCHAR(30) NOT NULL UNIQUE,

coach\_fname VARCHAR(15),

coach\_lname VARCHAR(15),

coach\_ID INT NOT NULL UNIQUE,

PRIMARY KEY(coach\_ID),

FOREIGN KEY(team\_name) REFERENCES TEAM(team\_name)

);

CREATE TABLE OFFENSIVE\_RANKING (

team\_name VARCHAR(30) NOT NULL UNIQUE,

total\_score INT,

yards\_per\_play DECIMAL(3,2),

total\_yards SMALLINT,

turnovers INT,

penalties INT,

PRIMARY KEY(total\_score, team\_name),

FOREIGN KEY(team\_name) REFERENCES TEAM(team\_name)

);

CREATE TABLE DEFENSIVE\_RANKING (

team\_name VARCHAR(30) NOT NULL UNIQUE,

total\_score INT,

yards\_per\_play DECIMAL(3,2),

total\_yards SMALLINT,

turnovers INT,

penalties INT,

PRIMARY KEY(total\_score, team\_name),

FOREIGN KEY(team\_name) REFERENCES TEAM(team\_name)

);

CREATE TABLE GAME (

game\_ID INT NOT NULL UNIQUE,

stadium\_ID INT NOT NULL,

home\_team\_name VARCHAR(30) NOT NULL,

away\_team\_name VARCHAR(30) NOT NULL,

final\_score VARCHAR(6) NOT NULL,

PRIMARY KEY (game\_ID),

FOREIGN KEY (stadium\_ID) REFERENCES STADIUM(stadium\_ID),

FOREIGN KEY (home\_team\_name) REFERENCES TEAM(team\_name),

FOREIGN KEY (away\_team\_name) REFERENCES TEAM(team\_name)

);

CREATE TABLE PARTICIPATES\_IN (

home\_team\_name VARCHAR(30) NOT NULL,

away\_team\_name VARCHAR(30) NOT NULL,

game\_ID INT NOT NULL UNIQUE,

game\_date DATE,

PRIMARY KEY(home\_team\_name, away\_team\_name, game\_ID),

FOREIGN KEY (home\_team\_name) REFERENCES TEAM(team\_name),

FOREIGN KEY (away\_team\_name) REFERENCES TEAM(team\_name),

FOREIGN KEY (game\_ID) REFERENCES GAME(game\_ID)

);

# 

# **8. Table Inserts**

This section contains example inserts of each table in the NFL database.

### Table TEAM inserts

INSERT INTO team(team\_name, state\_represented) VALUES('Seattle Seahawks', 'Washington');

### Table STADIUM inserts

INSERT INTO STADIUM(stadium\_ID, stadium\_name, location\_state, location\_city)

VALUES(

1, 'Lambeau Field', 'Wisconsin', 'Green Bay'

);

### Table PLAYER inserts

INSERT INTO PLAYER(player\_ID, team\_name, player\_name, position)

VALUES(

1, 'Baltimore Ravens', 'Alaka Otaro', 'LB'

);

### Table COACH inserts

INSERT INTO COACH(team\_name, coach\_fname, coach\_lname, coach\_ID)

VALUES(

'Atlanta Falcons', 'Dan', 'Quinn', 2

);

### Table OFFENSIVE\_RANKING inserts

INSERT INTO OFFENSIVE\_RANKING(team\_name, total\_score, yards\_per\_play, total\_yards, turnovers, penalties)

VALUES(

'Atlanta Falcons', 191, 5.8, 3399, 16, 74

);

### Table DEFENSIVE\_RANKING inserts

INSERT INTO DEFENSIVE\_RANKING(team\_name, total\_score, yards\_per\_play, total\_yards, turnovers, penalties)

VALUES(

'Baltimore Ravens', 189, 4.3, 3.97, 14, 52

);

### Table GAME inserts

INSERT INTO GAME(game\_ID, stadium\_ID, home\_team\_name, away\_team\_name, final\_score)

VALUES(

2, 18, 'New England Patriots', 'Houston Texans', '20|27'

);

### Table PARTICIPATES\_IN inserts

INSERT INTO PARTICIPATES\_IN(home\_team\_name, away\_team\_name, game\_ID, game\_date)

VALUES(

'New England Patriots', 'Houston Texans', 2, '2018-09-09'

);

# 

# **9. Creat**ion **and Processing of Test Data**

The test data used to populate the NFL database was collected directly from [www.nfl.com](http://www.nfl.com). We copied all of the data that we needed for each table into an excel document and used <http://beautifytools.com/excel-to-sql-converter.php> to convert the excel document into MySQL inserts saved in a .SQL file. This saved us a huge amount of time, because our database consists of over 2,500 inserts. We initially implemented and populated the NFL database manually on a local host machine to make sure that all of our table creations and inserts work working correctly. After successful implementation of the database locally we connected Amazon Web Services Relation Database Base Hosting to MySQL Workbench and recreated an instance of the database on the cloud service. We used MySQL Workbench to create and populate the database on AWS RDB. There were some issues with MySQL Workbench in regards to handling commented lines in .sql documents. This caused the first insert in each file to have syntax errors, and required them to be reinserted manually.

# 

# **10. Documented Changes**

This section contains changes made to the NFL Database and it is corresponding artifacts, as well as a discussion on the reasoning behind the changes that have been made.

## **10.1 Reasoning** forC**hanges**

We decided to remove attributes from the following entities to reduce the complexity of the database. We found that including extra information would greatly increase the time we would need to spend to insert all the required data. Since our group wants to prioritize other aspects of the project such as constraint testing and AWS hosting, we decided to limit the number of inserts. When we created our proposal we underestimated how big the scope was. This could be because none of our group members had previous experience creating SQL databases. We found that removing some of the lesser needed attributes from our entities (including dropping some entities completely) was the best way for our group to stay on track with our goal.

## **10.2 List of** C**hanges**

Added group members names to document header

Underlined ‘team\_name’ foreign key in ‘COACH’ table in Figure 2: Relational Model Diagram

Completely removed the SCORE weak entity and incorporated the final score of games into the GAME entity.

Modified Figure 1: Entity-Relationship Diagram to reflect changes.

Modified Figure 2: Relational-Data Model to reflect changes.

Changed the source of data for database to [www.nfl.com](http://www.nfl.com) in tooling section.

Added MySQL workbench to tooling section.

List of attributes removed/changed/added from database:

* PLAYER
  + Added ‘SAF’ to position constraint to include the safety position.
  + Removed Weight attribute.
  + Removed Salary attribute.
  + Removed Birthday attribute.
  + Changed player\_name from composite to attribute to normal attribute.
  + Removed Jersey\_number attribute.
  + Removed Height attribute.
* OFFENSIVE\_RANKING
  + Removed First downs
  + Removed Passing yards
  + Removed Passing Attempts
  + Removed Rushing Attempts
  + Removed Rushing Yards
* DEFENSIVE\_RANKING
  + Removed First downs
  + Removed Passing yards
  + Removed Passing Attempts
  + Removed Rushing Attempts
  + Removed Rushing Yards
* COACH
  + Removed Salary attribute
  + Removed Coach type
* STADIUM
  + Removed Capacity attribute.
  + Removed Surface attribute.
  + Removed Roof attribute.

# 

# **11. Queries**

### Table 2: List of Query Statements and their Purpose

|  |  |
| --- | --- |
| **SQL Query Statement** | **Purpose** |
| SELECT stadium\_name as Stadium, location\_city AS City , location\_state AS State  FROM STADIUM; | Lists the locations of all NFL stadiums |
| SELECT coach\_fname, coach\_lname  FROM COACH  INNER JOIN TEAM  ON COACH.team\_name=TEAM.team\_name  WHERE TEAM.team\_name='Seattle Seahawks'; | Lists the names of all coaches on the Seattle Seahawks |
| SELECT TEAM.team\_name, OFFENSIVE\_RANKING.total\_score  FROM TEAM, OFFENSIVE\_RANKING  WHERE TEAM.team\_name=OFFENSIVE\_RANKING.team\_name AND OFFENSIVE\_RANKING.total\_score>=200; | Lists the names and offensive total scores of all teams with offensive total scores of 150 or more |
| SELECT player\_name  FROM PLAYER  WHERE position='QB'; | Lists the names of all Quarterbacks (QBs) in the NFL |
| SELECT TEAM.team\_name, DEFENSIVE\_RANKING.total\_score AS Defensive\_Score  FROM TEAM  INNER JOIN DEFENSIVE\_RANKING  ON TEAM.team\_name=DEFENSIVE\_RANKING.team\_name  ORDER BY DEFENSIVE\_RANKING.total\_score DESC; | Lists the names of all teams and their defensive total score ordered by highest score to lowest |
| SELECT TEAM.team\_name, OFFENSIVE\_RANKING.total\_yards  FROM TEAM  INNER JOIN OFFENSIVE\_RANKING  ON TEAM.team\_name=OFFENSIVE\_RANKING.team\_name  WHERE OFFENSIVE\_RANKING.total\_yards > 3200  ORDER BY OFFENSIVE\_RANKING.total\_yards DESC; | Lists the names of all teams and their total offensive yards for teams with over 3200 offensive total yards ordered by most total yards to least |
| SELECT PLAYER.player\_name AS Player  FROM PLAYER  WHERE PLAYER.player\_name LIKE 'B%'; | Lists the names of all players whose names start with a B |
| SELECT player\_name as Player  FROM PLAYER, TEAM, DEFENSIVE\_RANKING  WHERE TEAM.team\_name=PLAYER.team\_name AND TEAM.team\_name=DEFENSIVE\_RANKING.team\_name AND DEFENSIVE\_RANKING.total\_score in(  SELECT max(DEFENSIVE\_RANKING.total\_score)  FROM DEFENSIVE\_RANKING); | Lists the names of all players of the team with the best defensive ranking |
| SELECT GAME.home\_team\_name AS Home\_Team, GAME.away\_team\_name AS Away\_Team, GAME.final\_score AS Final\_Score  FROM GAME  INNER JOIN PARTICIPATES\_IN  ON GAME.game\_ID=PARTICIPATES\_IN.game\_ID  WHERE PARTICIPATES\_IN.game\_date='2018-09-09'; | Lists the home team, away team and final score of all games that happened on 2018-09-09 |
| SELECT TEAM.team\_name, coach\_fname, coach\_lname, OFFENSIVE\_RANKING.turnovers AS Offensive\_Turnovers, DEFENSIVE\_RANKING.turnovers AS Defensive\_turnovers  FROM TEAM, COACH, OFFENSIVE\_RANKING, DEFENSIVE\_RANKING  WHERE TEAM.team\_name=COACH.team\_name AND TEAM.team\_name=OFFENSIVE\_RANKING.team\_name AND TEAM.team\_name=DEFENSIVE\_RANKING.team\_name AND (OFFENSIVE\_RANKING.turnovers=(  SELECT max(OFFENSIVE\_RANKING.turnovers)  FROM OFFENSIVE\_RANKING  )  OR DEFENSIVE\_RANKING.turnovers=(  SELECT max(DEFENSIVE\_RANKING.turnovers)  FROM DEFENSIVE\_RANKING  )); | Lists the team name, coach name, number of offensive turnovers and number of defensive turnovers for the teams with the most offensive turnovers or the most defensive turnovers in the NFL |

# 

# **12. Normalization**

### Table 3: Functional Dependencies and Normal Forms

|  |  |
| --- | --- |
| **Table and Functional Dependencies** | **Level of Normal Form and Possible Improvements (If Any)** |
| TEAM FD1: {team\_name}→{state\_represented} | 1NF: Yes   * All attributes are atomic   2NF: Yes   * All non-prime attributes are fully functionally dependent on the key   3NF: Yes   * team\_name is the superkey of the table   BCNF: Yes   * State\_represented is not a prime attribute so the clause is valid |
| STADIUM  FD1: {stadium\_ID}→{stadium\_name, location\_state, location\_city}  FD2: {location\_state}→{location\_city} | 1NF: Yes   * All attributes are atomic   2NF: Yes   * All non-prime attributes are fully functionally dependent on the key   3NF: No   * location\_city is dependent on the location\_state as there may be multiple cities across different states that have the same name, thus location\_city is transitively dependent on the original table   BCNF: No, 3NF violated  A possible normalization would split this table up so that location\_city dependent on location\_state would be its own table like in FD2, which would help us avoid anomalies in the future if two distinct stadiums have the same city name. |
| PLAYER  FD1: {player\_ID}→{team\_name, player\_name, position} | 1NF: Yes   * All attributes are atomic   2NF: Yes   * All non-prime attributes are fully functionally dependent on the key   3NF: Yes   * player\_ID is the superkey of the table   BCNF: Yes   * Attributes on the right side of the FD are not prime attributes so the clause is valid |
| COACH  FD1: {coach\_ID}→{team\_name, coach\_fname, coach\_lname} | 1NF: Yes   * All attributes are atomic   2NF: Yes   * All non-prime attributes are fully functionally dependent on the key   3NF: Yes   * coach\_ID is the superkey of the table   BCNF: Yes   * Attributes on the right side of the FD are not prime attributes so the clause is valid |
| OFFENSIVE RANKING  FD1: {total\_score, team\_name}→{yards\_per\_play, turnovers, penalties} | 1NF: Yes   * All attributes are atomic   2NF: Yes   * All non-prime attributes are fully functionally dependent on the key   3NF: Yes   * total\_score, team\_name is the superkey of the table   BCNF: Yes   * Attributes on the right side of the FD are not prime attributes so the clause is valid |
| DEFENSIVE RANKING  FD1: {total\_score, team\_name}→{yards\_per\_play, turnovers, penalties} | 1NF: Yes   * All attributes are atomic   2NF: Yes   * All non-prime attributes are fully functionally dependent on the key   3NF: Yes   * total\_score, team\_name is the superkey of the table   BCNF: Yes   * Attributes on the right side of the FD are not prime attributes so the clause is valid |
| GAME  FD1: {game\_ID}→{stadium\_ID, home\_team\_name, away\_team\_name, final\_score} | 1NF: Yes   * All attributes are atomic   2NF: Yes   * All non-prime attributes are fully functionally dependent on the key   3NF: Yes   * game\_ID is the superkey of the table   BCNF: Yes   * Attributes on the right side of the FD are not prime attributes so the clause is valid |
| PARTICIPATES\_IN  FD1: {game\_ID}→{home\_team\_name, away\_team\_name, game\_date} | 1NF: Yes   * All attributes are atomic   2NF: No   * game\_date is only fully functionally dependent on the game\_ID although the key is made up of game\_ID, home\_team\_name and away\_team\_name   3NF: No, 2NF violated  BCNF: No, 2NF violated   * In order to normalize this table to 3NF and BCNF we could modify the primary key to be only game\_ID. Making attributes on the right side of the FD are not prime attributes so the clause is valid. |

# 

# **13. Discussion**

## **13.1 Evaluation of the NFL** D**atabase** P**roject**

As a group we spent an average 10-20 hours a week working on the project. The largest amount of work was populating the database with the data that we acquired from [www.nfl.com](http://www.nfl.com) , mainly because we had over 2,500 inserts and needed to format the data before converting them to SQL queries and check for anomalies in the data due to manual data entry.

As a team there were many things that went right. We had clear, timely, and concise communication between all of our team members. This was mainly due to coordinating through discord and google documents. It allowed us to collaboratively work from remote locations and discuss the project with ease and at our own convenience. We also allocated responsibilities for completing iterations and milestones to people who had prior knowledge or were strong in those areas or volunteered to tackle said responsibilities.

There were some issues when it came to the scope of the project. Originally we had an unnecessarily large amount of attributes that caused us to have to restrict the scope of the database as a whole in order to prevent the team from spending a ridiculous amount of time inputting data. We also ran into some issues when hosting our database on AWS. The main issue with AWS was correctly configuring the security protocols. It took time to figure out how to make the database public and accessible from any IP address.

If we had another week, we would spend that time optimizing the UI and incorporating more statistical data on individual players in the NFL. The main reason we didn't include this in our scope was due to the scale of data collection involved and the time frame available to us.

## **13.2 Feedback from** D**ocument** I**teration** Su**bmission**s

The feedback from Iterations was minimal. The only suggestions made by the grader were to fix our Relational Model Diagram by underlining foreign key ‘team\_name’ in the ‘COACH’ table and include the group member names in the documentation. We fixed the anomalies in our documentation as soon as we were notified of them. As a team we were fortunate enough to take the time of producing iterations of our documentation that were of high quality and catch any mistakes we found before any formal submission deadlines.

## **13.3 Generation of** D**ata,** P**opulation of database, and** T**esting**

The test data used to populate the NFL database was collected directly from [www.nfl.com](http://www.nfl.com). We copied all of the data that we needed for each table into an excel document and used <http://beautifytools.com/excel-to-sql-converter.php> to convert the excel document into MySQL inserts saved in a .SQL file. This saved us a huge amount of time, because our database consists of over 2,500 inserts.

We initially implemented and populated the NFL database manually on a local host machine to make sure that all of our table creations and inserts work working correctly. After successful implementation of the database locally we connected Amazon Web Services Relation Database Base Hosting to MySQL Workbench and recreated an instance of the database on the cloud service.

We used MySQL Workbench to create and populate the database on AWS RDB. There were some issues with MySQL Workbench in regards to handling commented lines in .sql documents. This caused the first insert in each file to have syntax errors, and required them to be reinserted manually. To verify that all of our data was correctly stored in the Database we ran multiple queries, including the queries provided in section 11, and manually inspected the data for anomalies. Whenever we developed queries for testing we would make sure that we knew what the correct resulting output would be before running them on the NFL Database and made sure that the queries would access all the tables in our database as well as all the attributes in each table.

# 

# **14. Citations**

“MySQL Documentation.” *MySQL*, 2019, <https://dev.mysql.com/doc/>.

“RDS.” *Amazon*, Amazon Relational Database Service Documentation, 2019, <https://docs.aws.amazon.com/rds/index.html>.

“Official Site of the National Football League.” *NFL - Official Site of the National Football League*, 2019, [www.nfl.com/](http://www.nfl.com/).

“Excel to SQL converter” *Beautifytools, 2019,* http://beautifytools.com/excel-to-sql-converter.php.