

NFL Database Application

Team Members:

Chuck Zumbaugh

James Chapman

Vishnu Bondalakunta

Table of Contents

1. [Introduction](#)
2. [Database Implementation](#)
 - a. [Technical Description](#)
 - b. [Features](#)
 - c. [E/R Diagram and relational models](#)
 - d. [Queries](#)
3. [System Implementation](#)
4. [System Features and Usage](#)
 - a. [The Team Command](#)
 - b. [The Athlete Command](#)
 - c. [The Venue Command](#)
 - d. [The Game Command](#)
 - e. [The Top Comeback Wins Command](#)
 - f. [The Win Probability Command](#)
 - g. [The Save Command](#)
 - h. [The User Command](#)
 - i. [The Login Command](#)
 - j. [The Register Command](#)
5. [Evaluation](#)
6. [Technical Details and Justification](#)
7. [Summary and Discussion](#)
 - a. [General Summary](#)
 - b. [Learnings and Future Changes](#)
 - c. [Possible Improvements and Future Direction](#)
8. [Team Work Division and Experience](#)

Introduction:

The National Football League (NFL) is one of the major professional sports leagues in America and the highest professional level of football in the world. Sports databases, especially those for the NFL, store comprehensive collections of structured data related to various aspects of football. These databases serve multiple purposes, including statistical analysis, historical reference, betting odds calculation, fantasy sports management and more. They are commonly employed in sports apps to provide users with information related to various statistics. For this project, we designed a simple NFL database application that allows users to search for different information pertaining to the NFL.

The NFL Database Application manages and reports information for NFL games during the past decade (2013 - 2023). This system allows the user to easily retrieve information about players, teams, games, and even plays within each game (where available). With over 1.5 million records, users can obtain information down to each play made by a specific player in a game. The application itself is a lightweight, command line program that can be used once a user registers for an account. Target users are statisticians and eager NFL fans who want to have specific NFL game information at their fingertips! In addition to being able to view a data snapshot, users can export the snapshot in a filetype of their choice (ex. markdown or CSV) for their own use.

Database Implementation

Technical Description

The NFL Database stores detailed information regarding NFL teams, players, games, venues, plays, positions, and scores within the games. Additionally, the database supports the creation, updates, and deletion of user accounts for user tracking (and possibly billing) purposes. The system is implemented as a relational database in PostgreSQL with a number of requirements, as detailed below.

Users are identified by a uid and have attributes username, password, first name, last name, and created on.

- Users can favorite zero or one team
- Users can favorite zero or one athlete

Teams will be identified by a team name and have attributes location, abbreviation, primary color, and secondary color.

Athletes will be identified by an athlete ID and have attributes date of birth, birth city, birth state, first name, last name, height, and weight.

- A player can play for many teams in their career, and a team can have many players.

Each position will be identified by a position name and have attributes abbreviation and platoon(offense, defense, or special teams).

A player is contracted to play for a certain team in a specific position, and the contract lasts from the start date to the end date.

Venues will be identified by a name and have attributes capacity, city, state, grass (boolean), and indoor(boolean).

- A venue can have many home teams (ex. MetLife Stadium), but a team must have exactly one venue.

Games will be identified by a game ID and have attributes attendance, date, and utc_time.

- Many games can be played at a given venue, but each game must be played at exactly one venue.

The NFL schedule will be identified by the date and have attributes season_year, season_type, and week.

- Many games may be played on a given date, but each game must occur on exactly one date.
- Each game must have exactly one home team and one away team. Each team can play in many games.

Scores in each quarter (linescores) will be identified by the game its played in, the team it corresponds to, and quarter and have attributes score.

- Games and teams can have many linescores.

Each play will be identified by a play ID and have attributes play type, play text, quarter, seconds remaining in quarter, score value, start down, and end down.

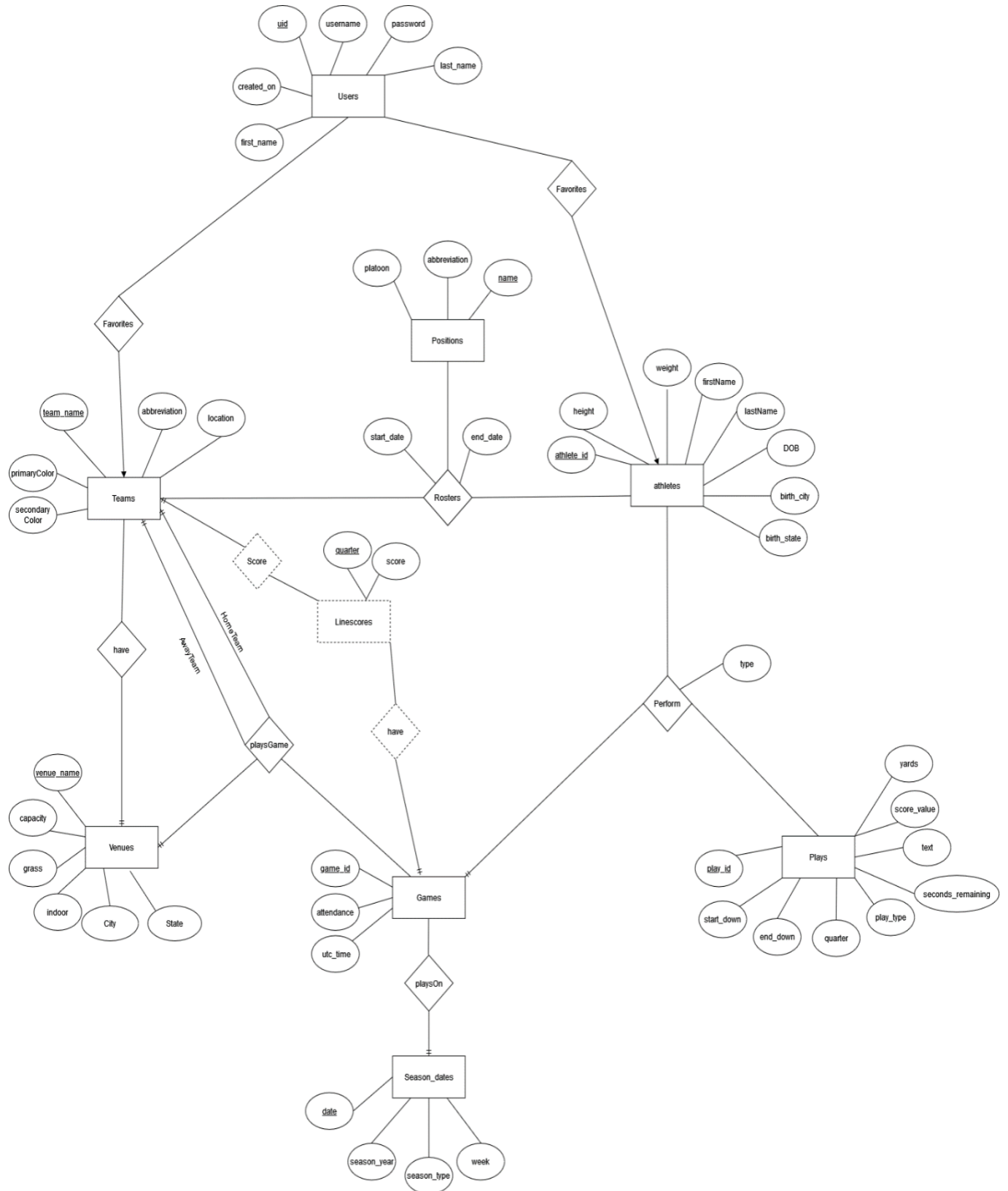
- A game can have many plays, and each player can be associated with many plays in each game. Additionally, a play can involve many players.

Features Implemented and How They Work

The primary purpose of the application is to provide detailed data regarding NFL games to users on request. As such, the primary feature is the ability to generate detailed reports and answers to questions such as game information, play-by-play data, player statistics, venue statistics, and the probability of a team winning a given game, among others. As users do not have detailed knowledge of in-game data, they are not permitted to modify or delete this data. However, they are free to update their profile. For example, a user may change his/her name or password at any time. In many cases a user will have a favorite team or athlete that they wish to follow. The system allows them to set a favorite team and athlete to indicate this. For convenience, the system will display the user's favorite player information and a summary of their favorite team's current record for the latest season (wins and losses, broken down by home and away wins). A user need not have a favorite team/athlete and may delete this at any time.

In some cases, user's may wish to utilize the data they obtain through this application for other purposes (ex. Machine learning, displaying on a website, etc.). To facilitate this, the data retrieved from various queries can be written to a file and saved to the user's device.

ER Diagram and Relational Schema:



Users(uid, username, password, first_name, last_name, created_on, favorite_team_name, favorite_athlete_id)

- favorite_team_name is a foreign key referencing Teams.team_name
- favorite_athlete_id is a foreign key referencing Athletes.athlete_id
- username is a unique key

Teams(team_name, location, abbreviation, venue_name, primary_color, secondary_color)

- venue_name is a foreign key referencing Venues.venue_name

Venues(venue_name, capacity, city, state, grass, indoor)

Games(game_id, date, attendance, home_team_name, away_team_name, venue_name, utc_time)

- date is a foreign key referencing Season_dates.date
- home_team_name is a foreign key referencing Teams.team_name
- away_team_name is a foreign key referencing Teams.team_name
- venue_name is a foreign key referencing Venues.venue_name
- date, home_team_name, away_team_name is a unique key

Season_dates(date, season_year, season_type, week)

Athletes(athlete_id, first_name, last_name, dob, height, weight, birth_city, birth_state)

Positions(position_name, abbreviation, platoon)

- abbreviation is a unique key

Rosters(team_name, athlete_id, position_name, start_date, end_date)

- team_name is a foreign key referencing teams.team_name
- athlete_id is a foreign key referencing players.player_id
- position_name is a foreign key referencing positions.position_name

Linescores(team_name, game_id, quarter, score)

- team_name is a foreign key referencing teams.team_name
- game_id is a foreign key referencing games.game_id

Plays(play_id, quarter, yards, score_value, play_type, text, seconds_remaining, start_down, end_down)

Player_Plays(play_id, player_id, game_id, type)

- play_id is a foreign key referencing plays.play_id
- player_id is a foreign key referencing Athletes.athlete_id
- game_id is a foreign key referencing Games.game_id

Views:

All_final_game_scores(game_id, home_team_name, home_team_score, away_team_name, away_team_score, home_winner_bool)

All_third_quarter_scores(game_id, home_team_name, home_team_score, away_team_name, away_team_score, away_team_winning_bool)

SQL Queries:

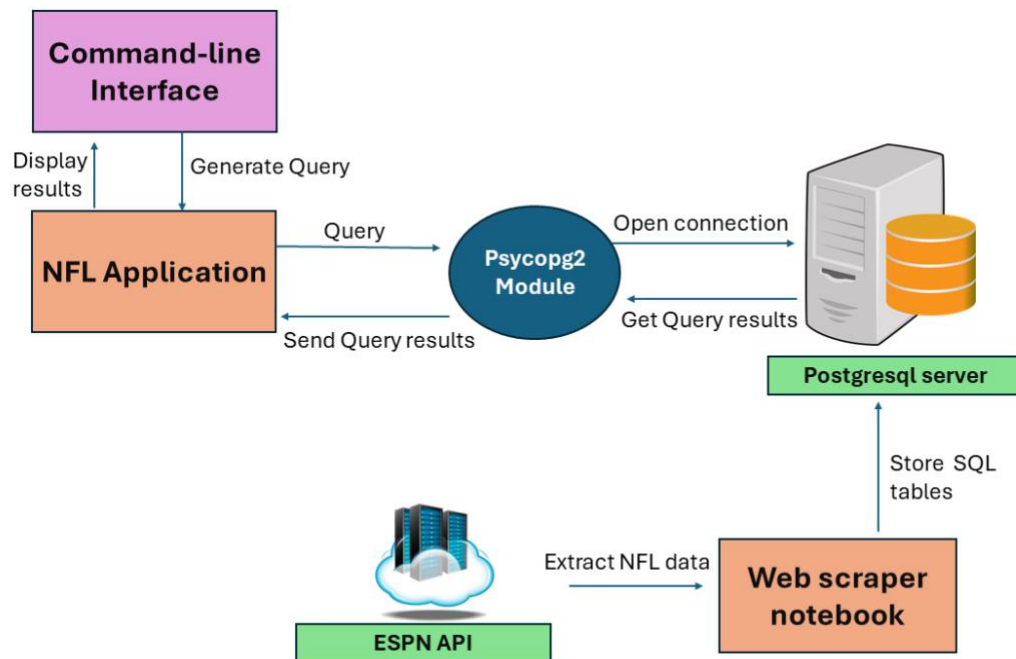
We implemented several interesting queries in our application. We have several queries implemented which are expected for such an app (For Ex: Finding specific games, players). We also have additional queries which return some interesting statistical information. We list the queries below.

- 1.) Weekly Receiving stats: For a given week, list the players (with receiving stats) and the number of receiving yards.
- 2.) Athlete Receiving stats: For a given athlete, list the number of receiving yards over all season weeks.
- 3.) Post season game count: For each team, list the number of post season games and the type (Wildcard, Divisional, Conference, Superbowl).
- 4.) Avg points grass indoor: How impactful (w.r.t total points) is playing on grass field versus turf and playing indoors versus playing outdoors?
- 5.) Team rivals: Find all the games and information which involves two particular teams playing each other.
- 6.) Top comeback wins: Which teams have the greatest number of 4th quarter comeback wins?
- 7.) Win probability: What is the probability that a given team will win given their 3rd quarter scores?
- 8.) First quarter greatest: Find all games where the total number of points scored in the first quarter is greater than the total number of points scored in any other quarter.
- 9.) Game recap: Find all the information about a requested game including scores and winner.
- 10.) Most home wins: Which stadium(s) has the most home wins in a given season year (has the greatest home-field advantage)?
- 11.) Passing Statistics: Find passer statistics such as Passing yards, Pass Attempts, Touchdown Passes, pass completions and passer rating for a given season year.
- 12.) Stadium occupancy: How full was a stadium for a given game?
- 13.) Player plays: Find the plays made by a player in a particular game.
- 14.) Player roster: What team and position does a given athlete play for currently?
- 15.) Game scores: Find the teams and scores for each game in a given season year and week.
- 16.) Game leaders: Find the passing, receiving, and rushing leaders for a given game
- 17.) Win loss: What is the number of games won and lost by a team in a given season?
- 18.) Venues: What is the venue(s) along with the home team that match with the searched keywords?
- 19.) Users: Find the given user's information.

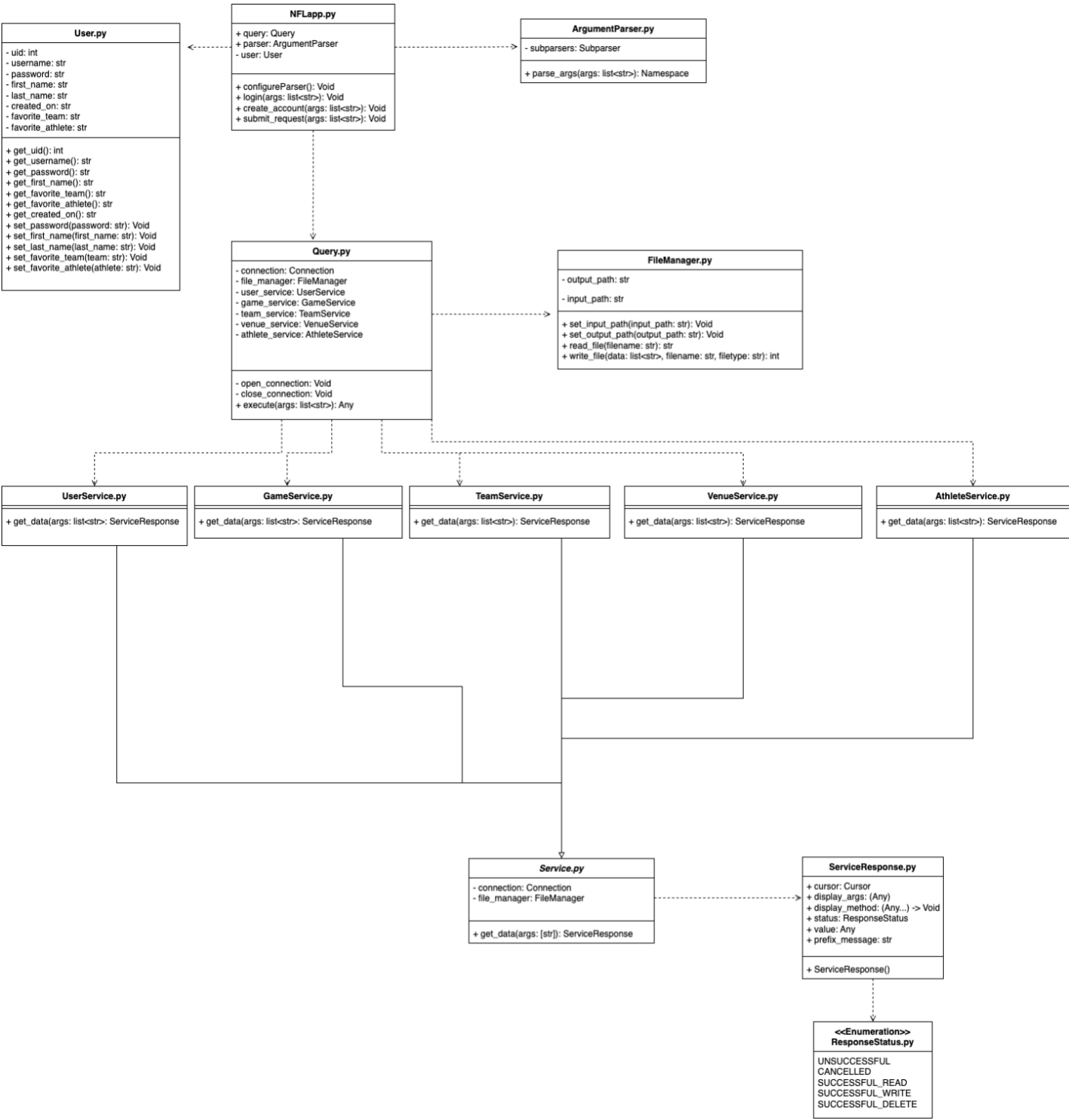
System Implementation:

Our application is implemented using the Python3 programming language and the PostgreSQL database management system. For database operations we use PLpgSQL. We also implemented a web scraper which extracts and processes information from an external ESPN API to populate our database. We used a Jupyter notebook for the webscraper implementation. After we have extracted the data as csv files from the ESPN API, we store them as tables in a PostgreSQL database on the KSU CS server. Our application makes a connection to this database using the psycopg2 python module when it is run.

Users can connect to our NFL application, register for an account and run queries on the database and save any results in either csv or md files.



System Class Diagram



System Features and Usage

The NFL Database Application is a command line program, that once launched, allows the user to issue commands to retrieve data. The commands are discussed in detail below, and each command can be controlled by the use of flags.

Top Level Commands

There are 10 top level commands that can be executed:

1. Team
2. Athlete
3. Venue
4. Game
5. Top_Comeback_Wins
6. Win_probability
7. Save
8. Login
9. Register
10. User

Each of these commands requires additional arguments and flags to retrieve specific data.

The Team Command

The Team command is used to retrieve information related to a team. The team command can be executed without any arguments or flags to retrieve information on all 32 teams in the NFL. Optionally, users can specify a team name to retrieve information for that team only.

Required Arguments

None

Flags and Optional Arguments

`-y or --year <year>`

This is used when retrieving the records for teams in a given season. When only this flag is used, the records of all teams during that season will be returned.

`-t or --team <team_name>`

This is used in conjunction with the `-y` or `--year` flag. When provided, only the record for the provided team name in the given season is returned.

`-psc or --postseason_count`

Get post-season game counts for each team. This calculates appearances in each of these games, not wins.

Usage

- > Team Returns all teams in the NFL
- > Team Chiefs Returns information on the Chiefs
- > Team -y 2020 Returns the records for all teams in the NFL during the 2020 season
- > Team -y 2020 -t Chiefs Returns the record for the Chiefs in the 2020 season
- > Team -psc Return the postseason game counts for each team across all years

```
> Team Chiefs
Name: Chiefs
Abbreviation: KC
Location: Kansas City
Home Stadium: GEHA Field at Arrowhead Stadium

Welcome, Chuck!
Favorite Team: Chiefs
Favorite Athlete: None
*** Please enter one of the following commands ***
> Build Database
> Team [<team_name>]
> Athlete <athlete_name> [-i]
> Venue [<venue_name>]
> Game -g <game_id> [-y <year>]
> Top_Comeback_Wins [-y <year>]
> Win_probability -t <team_name> -s <team_score> -op <opponent_score>
> Save <type> [-o <filename>]
> User [-f] [-t] [-a] [-d]
> quit
> []
```

Example use of the Team command to find information on the Chiefs. The output is colored in the teams colors.

The Athlete Command

The Athlete command is used to retrieve information for a given athlete. This command requires at least one argument, either the first or last name of the athlete.

Required Arguments

Either the first or last name of the athlete must directly follow the Athlete command. The default strategy is to search by first name.

Flags and Optional Arguments

`-l or --last`

Search by last name instead of first

`-S or --statistic`

Get receiving statistics for an athlete

`-y or --year <week>`

Specify the year to get weekly receiving statistics for

`-w or --week <week>`

Specify the week to get weekly receiving statistics for

`-st or --season_type <season_type>`

Specify the season type to get the weekly receiving statistics for. Accepted values are rs for regular season and ps for post season.

`-pr or --passer_rating`

Get passing statistics, including the passer rating for quarterbacks in a given year. Requires the use of the `-y or --year <year>` flag to specify the year. Optionally, the user can specify the athlete ID with the `-a or --athlete <athlete_id>` flag to get the values related to that athlete alone.

Usage

> Athlete Patrick Returns all athletes with the first name of Patrick

> Athlete Mahomes -l Returns all athletes with the last name of Mahomes

> Athlete -a 3116406 -S Get weekly receiving statistics for the athlete with an ID of 3116406

> Athlete -S -y 2023 -w 10 -st rs Get receiving statistics for all athletes during week 10 of the 2023 regular season.

> Athlete -pr -y 2020 Get passing statistics and passer ratings for all quarterbacks in 2020

> Athlete -pr -y 2020 -a 2330 Get passing statistics for athlete 2330 in 2020

```
> Athlete Mahomes -l

ID: 3139477
Name: Patrick Mahomes
Date of Birth: 1995-09-17
Height, in: 74
Weight, lbs: 225
Birth Place: Whitehouse, TX
Team Name: Chiefs
Position: Quarterback
Platoon: Offense

Welcome, Chuck!
Favorite Team: Chiefs
Favorite Athlete: None
*** Please enter one of the following commands ***
> Build_Database
> Team [<team_name>]
> Athlete <athlete_name> [-l]
> Venue [<venue_name>]
> Game -g <game_id> [-y <year>]
> Top_Comeback_Wins [-y <year>]
> Win_probability -t <team_name> -s <team_score> -op <opponent_score>
> Save <type> [-o <filename>]
> User [-f] [-t] [-a] [-d]
> quit
>
```

Example of using the Athlete command to find information on Patrick Mahomes.

The Venue Command

The Venue command is used to retrieve information related to venues (or stadiums). This can be executed with no arguments, in which case it returns all venues used by the NFL. Optionally, the user can specify the venue name (or a substring of the name).

Required Arguments

None

Flags and Optional Arguments

`-y` or `--year <year>`

If a year is specified, the venues with the greatest number of home wins, their teams, and the number of wins in that season will be returned.

`-S` or `--statistics`

Get the average points scored for various stadium and field combinations.

Usage

- > Venue Returns all venues
- > Venue GEHA Returns all venues with 'GEHA' in the name
- > Venue Field Returns all venues with 'Field' in the name
- > Venue -y 2023 Returns venues with the greatest number of home wins in 2023
- > Venue -S Returns the average points scored for combinations of grass/turf fields and indoor/outdoor stadiums

```
> Venue GEHA

Name: GEHA Field at Arrowhead Stadium
Home Team: Chiefs
Capacity: 76416
City: Kansas City
State: MO
Grass: True
Indoor: False

Welcome, Chuck!
Favorite Team: Chiefs
Favorite Athlete: None
*** Please enter one of the following commands ***
> Build_Database
> Team [<team_name>]
> Athlete <athlete_name> [-l]
> Venue [<venue_name>]
> Game -g <game_id> [-y <year>]
> Top_Comeback_Wins [-y <year>]
> Win_probability -t <team_name> -s <team_score> -op <opponent_score>
> Save <type> [-o <filename>]
> User [-f] [-t] [-a] [-d]
> quit
>
```

Example of using the Venue command to find information on GEHA Field at Arrowhead Stadium.

The Game Command

The Game command is one of the more flexible and powerful commands in the program. It is used to retrieve all information related to a given game or games. A large number of flags and arguments can be given to the Game command to control its execution.

Required Arguments and Flags

At the base level, at least one of the following flags and arguments must be provided. However, these may or may not be required if additional flags are used.

`-y <year>`

Returns information for all games in the given season

`-g <game_id>`

Returns information related to the game with the specified ID

Flags and Optional Arguments

The following flags and their arguments can be provided.

`-s or --score`

This specifies that you would like to retrieve only the scores for the games. This flag requires no additional arguments, but requires the user to specify what year and week of the season to search for scores in. This is done through the `-y <year>` and `-w <week>` flags.

`-w or --week <week>`

Specify which week of the season to search in.

`-p or --plays`

Tell the program that you wish to find plays in a given game. This flag accepts no arguments, but requires the use of the `-g --game <game_id>` and `-a or --athlete <athlete_id>` flag.

When used, the program will return all plays made by the given athlete in that game.

`-a or --athlete <athlete_id>`

Specify which athlete to find plays for. This flag accepts an athlete ID following the flag.

`-t or --team <team_name>`

Specify the name of the team to search for. This flag is used when retrieving the information related to games between two teams. If this flag is used, the opposing team must be specified using the `-op or --opponent <op_team_name>`.

`-pf or --percent_filled`

Specify that you would like to know how full the stadium was for a given game. This flag takes no arguments, but requires you to specify which game to compute the statistic for through the `-g or --game <game_id>` flag.

`-S` or `--statistics`

Specify that you would like to retrieve the leaders for passing, rushing, and receiving yards in the given game. This flag takes no arguments, but requires you to specify which game to compute the statistic for through the `-g` or `--game <game_id>` flag.

`-fq` or `--first_quarter`

Get games in which the total points scored in the first quarter was greater than any other quarter. When specified alone, this will return games that meet this criteria across all years. When the `-y <year>` flag is used, only the games in the specified year are returned.

Usage

The general use pattern for each of the features is listed below.

> `Game -y 2020` Return information for all games in the 2020 season

> `Game -g 401437927` Return information for the game with the id of '401437927'

> `Game -s -y 2020 -w 10` Return the teams and their scores for all games in week 10 of the 2020 season

> `Game -p -a 3139477 -g 401547235` Return all plays made by athlete with ID of '3139477' in the game with an ID of '401547235'

> `Game -g 401547235 -pf` Return the percent fill of the stadium for the game with ID of '401547235'

> `Game -t Chiefs -op Raiders` Return all games played by the Chiefs and Raiders in the database

> `Game -t Chiefs -op Raiders -y 2020` Return all games played by the Chiefs and Raiders in the 2020 season

> `Game -t Chiefs -op Chiefs` Return all games played by the Chiefs against any opponent in the database

> `Game -t Chiefs -op Chiefs -y 2020` Return all games played by the Chiefs against any opponent in the 2020 season

> `Game -g 401547235 -S` Return the leaders for passing, receiving, and rushing yards in the game with an ID of 401547235

> `Game -fq` Return all games in which the total points scored in the first quarter were greater than all other quarters

> `Game -fq -y 2023` Return all games in which the total points scored in the first quarter were greater than all other quarters in 2023

```
> Game -g 401547235

Game ID: 401547235
Date: 2023-12-31
Attendance: 73565
Home Team: Chiefs
Away Team: Bengals
Venue: GEHA Field at Arrowhead Stadium
Time: 21:25:00
Home Score: 25
Away Score: 17

Welcome, Chuck!
Favorite Team: Chiefs
Favorite Athlete: None
*** Please enter one of the following commands ***
> Build_Database
> Team [<team_name>]
> Athlete <athlete_name> [-i]
> Venue [<venue_name>]
> Game -g <game_id> [-y <year>]
> Top_Comeback_Wins [-y <year>]
> Win_probability -t <team_name> -s <team_score> -op <opponent_score>
> Save <type> [-o <filename>]
> User [-f] [-t] [-a] [-d]
> quit
> █
```

Example of using the Game command to find information of a game played by the Chiefs and Bengals. The output is colored in the winning team's colors (the Chiefs in this case).

The Top_Comeback_Wins Command

The Top_Comeback_Wins command displays the top 10 teams with the most comeback wins in a given year. A comeback win is defined as a team that is losing at the end of the 3rd quarter, but wins the game.

Required Arguments

-y <year>

Specify the year for which to search for comeback wins

Flags and Optional Arguments

None

Usage

> Top_Comeback_Wins -y 2023 Return the top 10 teams with the most comeback wins in 2023, along with their number of comebacks.

```
> Top_Comeback_Wins -y 2020
```

```
Team_name: Dolphins  
Comebacks: 3
```

```
Team_name: Lions  
Comebacks: 3
```

```
Team_name: Buccaneers  
Comebacks: 3
```

```
Team_name: Steelers  
Comebacks: 3
```

```
Team_name: Bears  
Comebacks: 3
```

```
Team_name: Cardinals  
Comebacks: 2
```

```
Team_name: Titans
```

Example of using the Top_Comeback_Wins command to find comebacks in 2020. Each team listed is colored in that team's colors.

The Win_Probability Command

The Win_probability command is used to determine a teams chance of winning given the team, their score in the third quarter, and their opponents score in the third quarter. This command utilized a logistic regression model to predict the chance of winning.

Required Arguments

`-t or --team <team>`

The team's name

`-s or --score <score>`

The teams score at the end of the third quarter

`-op or --opponent <score>`

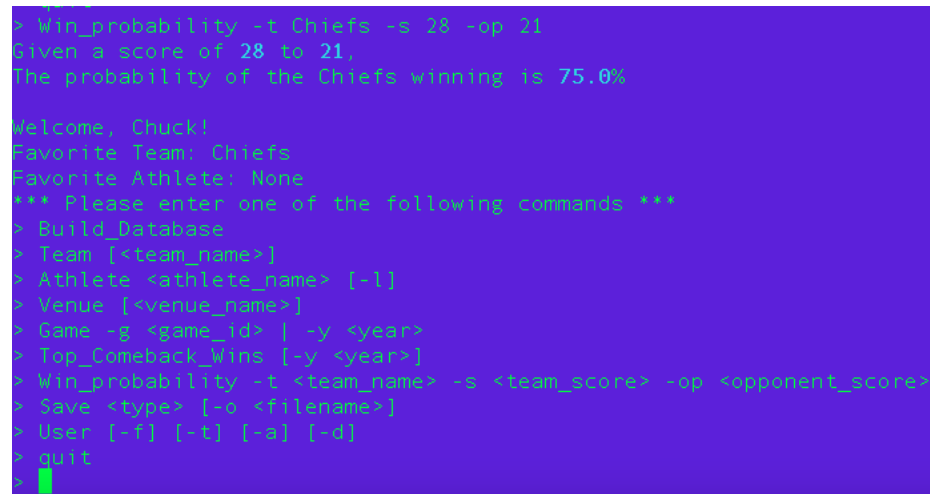
The opponents score at the end of the third quarter

Flags and Optional Arguments

None

Usage

`> Win_probability -t Chiefs -s 28 -op 21` Predict the Chief's chance of winning given a score of 28 and an opponent's score of 21.



```
> Win_probability -t Chiefs -s 28 -op 21
Given a score of 28 to 21.
The probability of the Chiefs winning is 75.0%

Welcome, Chuck!
Favorite Team: Chiefs
Favorite Athlete: None
*** Please enter one of the following commands ***
> Build_Database
> Team [<team_name>]
> Athlete <athlete_name> [-l]
> Venue [<venue_name>]
> Game -g <game_id> [-y <year>]
> Top_Comeback_Wins [-y <year>]
> Win_probability -t <team_name> -s <team_score> -op <opponent_score>
> Save <type> [-o <filename>]
> User [-f] [-t] [-a] [-d]
> quit
>
```

An example of using the Win_probability command to find the probability of the Chiefs winning a game with a score of 28 and an opponent score of 21 in the 3rd quarter.

The Save Command

In some cases, a user may wish to save the results returned by the program. This can be accomplished with the Save command. The Save command requires a filetype to be specified, and optionally a filename. The results of the most recently executed query will be saved in the user's Downloads folder.

Required Arguments

The filetype must immediately follow the Save command. Supported filetypes are markdown (specified as md) or comma separated value (specified as csv). The program will write the file in the appropriate format.

Flags and Optional Arguments

`-o` or `--output <filename>`

The name of the file, without the extension. If this is not provided, the default filename is NFL_last_data.

Usage

- > Save md Save the results of the last executed query as a markdown file
- > Save csv Save the results of the last executed query as a CSV file
- > Save md -o my_data Save the results of the last executed query as a markdown file with the filename of 'my_data'

player_id	name	passing_yard	pass_attemp	pass_comple	touchdown	passes_inter	passer_rating
3139477	Patrick Mahomes	5807	701	459	48	8	60.4
3122840	Deshaun Watson	5212	577	399	33	6	64.6
3918298	Josh Allen	4765	594	401	34	10	60.4
14881	Russell Wilson	4749	586	403	40	12	60.4
2330	Tom Brady	4716	611	395	36	9	56.3
8439	Aaron Rodgers	4642	569	400	48	6	60.4
14880	Kirk Cousins	4486	540	360	34	12	60.4
11237	Matt Ryan	4442	587	380	24	11	56.3
4038941	Justin Herbert	4182	561	370	28	10	56.3
5529	Philip Rivers	4061	516	352	23	9	56.3
3917315	Kyler Murray	4037	546	365	26	11	56.3
14876	Ryan Tannehill	3985	515	331	37	9	56.3
5536	Ben Roethlisler	3981	601	392	33	9	52.1
3046779	Jared Goff	3963	554	371	20	12	56.3
16757	Derek Carr	3887	482	322	25	5	60.4
12483	Matthew Stafford	3842	499	320	23	6	56.3
16728	Teddy Bridgewater	3623	476	328	15	9	56.3
3052587	Baker Mayfield	3349	453	283	25	7	56.3
3916387	Lamar Jackson	3054	422	266	24	9	56.3
2580	Drew Brees	3048	392	278	22	8	56.3
3915511	Joe Burrow	2779	402	261	13	6	52.1
3924327	Drew Lock	2767	400	226	14	14	52.1

An example of the CSV file output in Excel displaying passing statistics.

The User Command

The User command is used for user related services. This command is used to update data in the user's account, and can be used to delete the account if desired.

Required Arguments

While the User command does not require any arguments on it's own, arguments are expected for some of the various flags that are used to update and delete data. If nothing is given to the User command, the user's details are printed.

Flags and Optional Arguments

`-f or -favorite`

Specify that you would like to favorite either a team or athlete

`-t or --team <team_name>`

Used in combination with the `-f` flag to specify a team name to favorite

`-a or --athlete <athlete_id>`

Used in combination with the `-f` flag to specify an athlete to favorite.

`-d or -delete`

Specify that you would like to perform a delete operation. When used without other flags, this will prompt an account deletion. When combined with the `-f` flag and either `-t` or `-a`, specify that you would like to delete your favorite team or athlete.

`-U or --update <field>`

Specify that you would like to update the provided field. Accepted fields are `'first_name'`, `'last_name'`, or `'password'`

`-V or --update <value>`

Set the new value to use with the `-U` flag.

Usage

`> User` Show the user's details

`> User -f -t Chiefs` Set Chiefs as your favorite team

`> User -f -a 3139477` Set the athlete with an ID of 3139477 as your favorite

`> User -f -t -d` Delete your favorite team

- > User -f -a -d Delete your favorite athlete
- > User -d Delete your account. This is an irreversible action
- > User -U first_name -V John Change your first name to John
- > User -U last_name -V Smith Change your last name to Smith
- > User -U password -V securePassword Change your password to securePassword. You will not be logged out by this action, but you will need your new password during the next sign in.

```
Welcome, Chuck!
Favorite Team: Chargers
Favorite Athlete: None
*** Please enter one of the following commands ***
> Build_Database
> Team [-t <team_name>]
> Athlete [-a <athlete_name> [-l]]
> Venue [-v <venue_name>]
> Game [-g <game_id>] [-y <year>]
> Top_Comeback_Wins [-y <year>]
> Win_probability [-t <team_name> -s <team_score> -op <opponent_score>]
> Save <type> [-o <filename>]
> User [-f] [-t] [-a] [-d]
> quit
> User -U password -V myPass
password has been successfully updated

Welcome, Chuck!
Favorite Team: Chargers
Favorite Athlete: None
*** Please enter one of the following commands ***
> Build_Database
> Team [-t <team_name>]
> Athlete [-a <athlete_name> [-l]]
> Venue [-v <venue_name>]
> Game [-g <game_id>] [-y <year>]
> Top_Comeback_Wins [-y <year>]
> Win_probability [-t <team_name> -s <team_score> -op <opponent_score>]
> Save <type> [-o <filename>]
> User [-f] [-t] [-a] [-d]
> quit
>
```

An example of updating the user's password to 'myPass'

```

Welcome, Chuck!
Favorite Team: Chargers
Favorite Athlete: None
*** Please enter one of the following commands ***
> Build_Database
> Team [<team_name>]
> Athlete <athlete_name> [-l]
> Venue [<venue_name>]
> Game -g <game_id> | -y <year>
> Top_Comeback_Wins [-y <year>]
> Win_probability -t <team_name> -s <team_score> -op <opponent_score>
> Save <type> [-o <filename>]
> User [-f] [-t] [-a] [-d]
> quit
> User -f -t Chiefs
Team Chiefs has been successfully favorited

Welcome, Chuck!
Favorite Team: Chiefs
Favorite Athlete: None
*** Please enter one of the following commands ***
> Build_Database
> Team [<team_name>]
> Athlete <athlete_name> [-l]
> Venue [<venue_name>]
> Game -g <game_id> | -y <year>
> Top_Comeback_Wins [-y <year>]
> Win_probability -t <team_name> -s <team_score> -op <opponent_score>
> Save <type> [-o <filename>]
> User [-f] [-t] [-a] [-d]
> quit
>

```

An example of updating the user's favorite team

```

Welcome, Chuck!
Favorite Team: Chiefs
Favorite Athlete: None
*** Please enter one of the following commands ***
> Build_Database
> Team [<team_name>]
> Athlete <athlete_name> [-l]
> Venue [<venue_name>]
> Game -g <game_id> | -y <year>
> Top_Comeback_Wins [-y <year>]
> Win_probability -t <team_name> -s <team_score> -op <opponent_score>
> Save <type> [-o <filename>]
> User [-f] [-t] [-a] [-d]
> quit
> User -d
Are you sure you want to delete this account? (y/n)

```

An example of deleting a user's account, and the confirmation prompt shown

The Login Command

The login command is used to authenticate and gain access to the program.

Required Arguments and Flags

You must specify the username and password to use. This is done as follows.

```
-u or --username <username>
```

Provide the username

```
-p or --password <password>
```

Provide the password

Usage

```
> NFLapp Login -u <username> -p <password>
```

```
(base) chuckzumbaugh@ip-10-130-49-123 src % NFLapp Login -u czunbaugh -p myPass
Connection Established!
***Favorite Team Most Recent Record***

Team Name: Chiefs
Home Wins: 5
Home Losses: 4
Away Wins: 5
Away Losses: 2
Record: 10 W - 6 L
*****

Welcome, Chuck!
Favorite Team: Chiefs
Favorite Athlete: None
*** Please enter one of the following commands ***
> Build_Database
> Team [<team_name>]
> Athlete <athlete_name> [-l]
> Venue [<venue_name>]
> Game -g <game_id> | -y <year>
> Top_Comeback_Wins [-y <year>]
> Win_probability -t <team_name> -s <team_score> -op <opponent_score>
> Save <type> [-o <filename>]
> User [-f] [-t] [-a] [-d]
> quit
>
```

An example of using the Login command to sign in to the system. Note that the latest record of the user's favorite team is printed when the system starts.

The Register Command

The Register command is used to register an account. The system will prompt you to enter various details.

Required Arguments

None

Usage

> NFLapp Register Prompts to enter information will follow and you will need to provide a username, password, and name.

```
(base) chuckzumbaugh@ip-10-130-49-123 src % NFLapp Register
Connection Established!
Use the following steps to register for an account...
Please choose a username: my_username
Please choose a password: myPass
Please enter your first name: FirstName
Please enter your last name: LastName
Thank you for registering for an account
Your account was created successfully. Please log in.
(base) chuckzumbaugh@ip-10-130-49-123 src %
```

An example of registering an account with the system. The user is prompted to choose a username, password, first name, and last name.

Evaluation:

Performance

We evaluated the performance of our application by analyzing the time it took for the response to be returned to the user. For many of the queries involving smaller tables, the execution time was quite fast without additional optimization. However, queries involving larger tables, specifically the `plays` and `player_plays` tables, we had relatively slower execution time. We performed additional optimization through the use of indexes to improve performance, as discussed below.

Easiness to use

One of the major design principals we had when developing this application was to make it detailed, yet easy to use. Given the large number of queries spanning a large number of question types, this was one of the larger challenges we faced. We made a design decision to limit the number of top-level commands to facilitate a better user experience. This was accomplished by grouping queries involving logically similar data into a single command, and using flags and optional arguments to control the behavior of the command. While this decision requires users to pass flags and arguments into each command, it resulted in a simpler and more approachable interface. Additionally, we believe that the logical connections between the results returned by each command will aid in a broader understanding of how the application is used.

In addition to limiting the number of top-level commands, we created flags that were intuitive and consistent across the application. For example, the `-y` flag to specify a year is consistent across all commands. If a command accepts a `-y` flag, it controls the year and nothing else. This provides the user with a consistent experience across the application, as it would be confusing if `-y` controlled the year in one query but something else in another. There were some challenges with this, as some of the options we wanted to provide could in theory use the same short flag. For example, we have implemented flags to obtain scores and statistics in the application, which depending on the context could both utilize a `-s` flag. We remedied this by choosing `-s` to correspond to scores, and `-S` to correspond to statistics. While the mixed case flags may seem confusing at first, we believe that this is a better approach than using the same flag to control different options, and it is a common pattern across command line programs. We follow CLI best practices and display help text when the `--help` flag is used and report relevant errors. Additionally, our project is well documented and easy to navigate for either using the app or developing it further.

Use of indexes

We used the following indexes in our database:

```
CREATE INDEX idx_pp_play_id ON player_plays(play_id);
CREATE INDEX idx_pp_game_id ON player_plays(game_id);
CREATE INDEX idx_plays_play_id ON plays(play_id);

CREATE INDEX idx_athletes_first_name ON athletes(first_name);
```

```
CREATE INDEX idx_rosters_athlete_id ON rosters(athlete_id);
CREATE INDEX idx_linescores_game_id ON linescores(game_id);
```

The use of these indexes substantially sped up the response time of queries involving some of the larger tables in our database. Some data related to the performance of various queries is shown below. Since we do not allow users to modify data in any of the NFL data tables, the use of these will have a minimal impact on application performance.

Performance improvement using indexes

Query	Original Time	Time With Index
Games	9.602	0.415
Players	10.884	11.801
Statistics	59.195	17.559
Top5TeamAttendance	21.212	24.291
TopComebackWins	43.464	44.694
Venue	0.185	0.189
PercentFilled	0.199	0.232
HomeFieldAdvantage	47.462	46.354
TeamRivals	36.069	36.02
WeeklyReceivingStats	72.203	12.807
AthleteReceivingStats	68.044	64.269
FirstQuarterGreater	22.425	23.63
Passing	91.027	85.181
PlayerGamePlays	59.271	0.697
Scores	20.879	5.601
TeamRecords	54.099	52.299
TeamPostSeasonGameCount	3.789	0.769
AvgPtsGrassIndoor	33.052	33.075
Users	0.142	0.115
Win probability	67.931	66.235

Technical Details and Justification

- 1.) The PostgreSQL database management system is used in this project. This DBMS was chosen because it 1) is a relational DBMS, 2) it provides a more correct SQL implementation than other options such as MySQL, 3) it is free, and 4) the KSU Computer Science department maintains a PostgreSQL server.
- 2.) The application is written in the Python programming language. In many cases, there are several languages that can be used to implement a given project, and this is no exception. For example, this application could have been written in C++, Java, and many others. We chose to use Python because it is commonly used for data science tasks, meshed well

with our data collection methods, and all team members were familiar with its syntax and use.

- 3.) Psycopg2 was used as the database connection module. This library is commonly used for connecting to PostgreSQL databases in Python, and was chosen because of its rich documentation, ease of use, and support for parameterized queries and prepared statements.
- 4.) The Python Argparse module is used to parse user commands. This is a common approach when designing command line programs using Python, and provides a relatively simple interface for implementing commands and flags. Rather than writing our own parser, this approach was used for speed of development and consistency with the Python ecosystem and developer community.
- 5.) PuTTY/Terminal color interface: Our application supports colored font display for terminal outputs. We use the python module 'rich' for this implementation because it provides a high-level interface to manipulate the appearance of the terminal and is well-documented. This is why certain queries are designed to also return jersey colors of teams so that we can display team-based results in a color code matching their actual jersey colors.

We opted for this approach as this would be easier than implementing a web interface but at the same time being more interactive than the usual terminal interface.

- 6.) Logistic Regression: For certain queries such as finding the winning probability of teams, we also train a simple logistic regression model based on the collected data. We used 'sklearn' module for this implementation. We have implemented the example query: Given the scores for the third quarter in a game, what is the probability that a given team would win?

During a live game, spectators are frequently curious about the chances of a team winning the game given the current situation. Also, live betting sites recalculate odds as the game progresses. Calculating live win-loss probability is a very common feature in sports which motivated us to include this functionality.

Summary and Discussion

General Summary

We implemented a user-friendly application that retrieves interesting statistics from NFL games. These statistics are of interest to various groups of people using sports apps, betting sites, fantasy sports etc. Our design is mostly back-end focused and deals with querying a database to generate statistics for some commonly expected information on games, although we do offer a basic interactive command line user interface.

Learnings:

We learnt a lot from working on this project. We encountered many unexpected situations when dealing with real data obtained from the web.

- 1.) For starters, we had to do additional processing such as adding and deleting exceptional entries to ensure the integrity of our data.
- 2.) We were also surprised when we discovered that certain expected functional dependencies did not hold due to some technicalities. For example, there were few players who had contracts with two different teams on the same day. This occurred because their old contract with one team ended on a particular day and their new contract started on the next day but was stored in the database using the previous day due to time zone differences.
- 3.) The data for older years such as 2013 was stored slightly differently. For example, the older data used 'pass' as a single play type while the new data used 'pass attempt', 'touchdown passes', 'pass interceptions'.
- 4.) The project enabled meeting and working with other people. Designing and discussing various ideas was an enjoyable experience.

Possible Improvements and Future Changes:

- 1.) Currently, the data stored in our database is static data over the past decade of NFL games. It would be interesting to have an application that regularly updates the database by automatically scraping the latest NFL data. However, there are challenges to face as the new data stored in ESPN website may not follow the same method and structure of storage(ESPN Data is stored as json files in big data platforms such as NoSQL databases).
- 2.) It would also be interesting to provide an interface to users with higher levels of access('admins') to add more queries or to modify existing queries to the application.
- 3.) Naturally, an interactive web interface is the next step and possibly deploying the application as an app on Android or iPhone platforms.
- 4.) The data in this database, and the NFL in general, is quite complex. There are many statistics and aggregation methods that are commonly used beyond what is implemented in this application. Since it is not feasible to implement all of this in the current application, we chose a subset of queries that were interesting to us, involved data in many tables, and answered relevant questions. Some future changes could be to extend this program to provide additional statistics and information to users. This is likely best done using a website or desktop/mobile application for a better user experience.

- 5.) Currently, passwords in the user table are stored as plain text. This is an obvious security concern, as bad actors who gain access to the system will have an easy time stealing user accounts. Future updates to this application should encrypt the passwords and store the encrypted password in the database.

Teamwork Division and experience:

We did not rigorously partition work and distributed amongst ourselves. We had regular meetings where we discussed things to accomplish for the next meeting or next assignment (Project proposal/SQL Queries).

Everyone equally contributed to design and discussion of various stages of implementation (Web scraping, Raw data processing, Database design, SQL queries, python implementation, user interface). While the contributions of each team member varied by the component of the project, each team member generally had an equal contribution over the entirety of the project.