1. **Explain the use case / Goal / Objective**
2. **Collect data. Explain why you choose this data, from these data sources**
   1. **Web Scraping**
   2. **API**
   3. **Flat File**
   4. **Database**
   5. **Big Data System**
3. **Clean data and execute exploratory data analysis (+vizualisation)**
4. **Choose the database type (compare several types and explain why)**
5. Create an entity-relationship diagram (at least 4 entities)
6. Create a database (database, tables)
7. Add data to the database
8. Create 5 scripts showing the insights
9. Prepare the 10 pages report (Can go over but don’t go too OTT)
10. Prepare presentation

## **Requirements & Deliverables**

Each student should upload all the project materials to Github.

You should deliver:

* **Planning of your project in Trello/Jira,**
* Code in Python for data collection and cleaning,
* ER model,
* data sources and metadata,
* database script,
* **report (10 pages)**
* slides.

The link to the Github repository and the report in pdf format should be sent should be shared with your teaching team by Tuesday of Week 9.

## **Presentation**

The presentation time limit is 20 minutes. You will have 20 minutes for Q&A.

The slides of your presentation could included but is not limited to the content listed below:

* Title of the project + Name
* Description of your Project (Planning, ER, Database Schema, Queries, Methodology)
* Challenges
* Process
* Highlights
* Main results
* Next steps
* Demo (If applicable)

Possible Data sources:

GCP - Public Datasets (For Big Data System)

<https://opendata.paris.fr/pages/home/>

<https://data.gov.uk/>

[https://data.iledefrance.fr/pages/home-open-data/  
https://kaggle.com](https://data.iledefrance.fr/pages/home-open-data/)

|  |  |
| --- | --- |
|  | Data Analytics |

Improving Fantasy Premier

League Score with Data Analytics

& Machine Learning

Oliver Skelley

Nov, 2023

**Table of content**

**Introduction**

The Premier League is the top league in the English football pyramid. It began in 1992, taking over from its predecessor ‘The Football League First Division’. Every year 3 clubs are relegated from the league into the Championship and 3 clubs from the Championship are promoted to the Premier League.

It is the highest grossing league in world football, it currently takes in around €6.5 billion of revenue a season with forecasts indicating that number will continue to rise. This is a massive amount of growth from its origins, taking just over €300 million in its first season. The increased monetization from all revenue streams such as tv rights and sponsorship deals has added more risk and reward to playing in the Premier League.

Changes in how clubs are operated have occurred across the league to accommodate for the new opportunities that the potential revenue allows. The salaries and valuation of personnel working for the football clubs have risen dramatically. In recent years there has been a growing reliance on data to help guide the decisions that all personnel must make. This data is used in all areas of football from deciding on which players to purchase in the transfer market to how the gambling industry decides what odds they will choose. The data that is collected has been increasing significantly with the help of artificial intelligence and tracking hardware. The global industry for the Sports Analytics Market is forecast to rise to $8.4 billion by 2026.

In 2002 the Fantasy Premier League (FPL) game was launched; it is an opportunity for all fans of the Premier League to create their own virtual team. As the available football data has improved so has the popularity of the game. Each player builds a team of 15 footballers, and they gain points when those footballers perform well. Each week the player must select 11 footballers to field in their team and they can make a limited number of transfers to change the players in their squad. I have played FPL on and off for a few years. I would like to improve my global ranking through this project by using Data Analytics and Machine Learning.

**Data and data sources**

The data I will be collecting for this project will be from two main sources: the Fantasy Premier League API/website/csv files and from the FBREF website.

**Fantasy Premier League (FPL):**

The FPL website has data available to players to help inform the choices they make each week. The data includes a variety of player performance statistics and the points they have scored based on a variety of in-game attributes. In recent years it has made expected statistics available. The expected statistics are incredibly powerful as they are more indicative of a player’s potential points than the actual underlying statistic that they are measuring. For example, if a footballer scores a goal in a game that may indicate that they are likely to score in the next game. However, if they score a goal from an ‘expected goal’ of 0.1 then it indicates that they were lucky to score that goal. This is incredibly important as luck is a massive contributor to the points scored by footballers each week but is unhelpful in deciding whether to include a footballer in the FPL squad. FPL only recently introduced the expected statistics and therefore the sample size of available data is relatively small.

The API that FPL makes available to the public is only for the current season of the Premier League. This posed a problem for my project as I couldn’t achieve enough statistically significant information to analyze or train a machine learning model with. Thankfully the previous season information has been harvested and stored online in a github repository: <https://github.com/vaastav/Fantasy-Premier-League>

Within this repository there is data relating to the seasons from 2016 to the current season. Every footballer in the league has data for all the matches that were played in each season.

**FPL Data Variables:**

|  |  |
| --- | --- |
| **name** | Player name |
| **position** | Player position in FPL |
| **team** | What team the player plays for |
| **xP** | Expected points based on performance in the game |
| **assists** | Number of passes provided that led to a goal |
| **bonus** | Number of bonus points awards from 0-3 based on players relative performance during the game |
| **bps** | Total number of points awarded to a player based on players relative performance during the game. |
| **clean\_sheets** | Did the team the player plays for concede a goal |
| **creativity** | Metric for how many goal creating actions occurred |
| **goals\_conceded** | Number of goals conceded by players team |
| **goals\_scored** | number of goals scored by player |
| **ict\_index** | Combined metric for creativity, influence and threat |
| **influence** | Metric for how much the player influenced the match |
| **minutes** | Minutes played by player in the match |
| **own\_goals** | Number of own goals scored by a player |
| **penalties\_missed** | Number of penalties scored by a player |
| **penalties\_saved** | Number of penalties saved by a player |
| **red\_cards** | Red cards awarded to player |
| **round** | What round of the league the match is based on (1-38) |
| **saves** | Number of saves made by a goalkeeper |
| **selected** | Number of virtual teams a player has been selected by prior to a particular game |
| **team\_a\_score** | Number of goals scored by the away team |
| **team\_h\_score** | Number of goals scored by the home team |
| **threat** | Metric for how much the player threatened to score a goal |
| **total\_points** | Total points received by a player that week |
| **transfers\_in** | Number of users who transferred a player into their team for that specific game week |
| **transfers\_out** | Number of users who transferred a player out of their team for that specific game week |
| **was\_home** | Was the game played at home or away from the players team  stadium |
| **yellow\_cards** | Yellow cards awarded to player |

**FBREF:**

FBREF is a website that uses has thousands of tables with millions of datapoints for hundreds of football leagues around the world. It has raw statistics for football matches but also has hundreds of calculated/engineered statistics for the football matches it records. It primarily uses ‘StatsBomb’, a sports data company, as its main data source. FBREF has access to an API with all of the data it needs. The data is filtered and published on the website. They are not allowed to provide the data in a downloadable format or as an API as they do not have the required license to provide that publicly. However, they allow web-scraping of the tables found on their website: <https://fbref.com/en/comps/9/2022-2023/stats/2022-2023-Premier-League-Stats>

The FBREF data is very useful as it has a lot of interesting variables that it records. The most important area of these variables is the expected statistics which are absent from the FPL data for earlier seasons. It also records more data relating to individual metrics. For example instead of just passes it records passes for a different variety of ranges.

**FBREF Data Variables:**

|  |  |
| --- | --- |
| **Date** | Date listed is local to the match |
| **Day** | Day of week |
| **Round** | Round or Phase of Competition |
| **Start** | In starting lineup, \* = squad captain |
| **Pos** | Position most commonly played by the player |
| **Min** | Minutes played by player in the match |
| **Gls** | Goals scored or allowed |
| **Ast** | Assists made |
| **PK** | Penalty Kicks Made |
| **PKatt** | Penalty Kicks Attempted |
| **Sh** | Shots Total, Does not include penalty kicks |
| **SoT** | Shots Total on Target, Does not include penalty kicks |
| **CrdY** | Yellow Cards |
| **CrdR** | Red Cards |
| **Touches** | Number of times a player touched the ball. Receiving a pass,  then dribbling, then sending a pass counts as one touch |
| **Tkl** | Number of players tackled |
| **Int** | Interceptions |
| **Blocks** | Number of times blocking the ball by standing in its path |
| **xG** | Expected Goals |
| **npxG** | Non-Penalty Expected Goals |
| **xAG** | Expected Assisted Goals |
| **SCA** | Shot-Creating Actions |
| **GCA** | Goal-Creating Actions |
| **Cmp** | Passes Completed |
| **Att** | Passes Attempted |
| **Cmp%** | Pass Completion Percentage |
| **PrgP** | Progressive Passes |
| **Carries** | Number of times the player controlled the ball with their feet |
| **PrgC** | Carries that move the ball towards the opponent's goal line |
| **Att. 1** | Number of attempts to take on defenders while dribbling |
| **Succ** | Number of defenders taken on successfully, by dribbling past them |
| **PassLive** | Completed live-ball passes that lead to a shot attempt |
| **PassDead** | Completed dead-ball passes that lead to a shot attempt. |
| **TO** | Successful take-ons that lead to a shot attempt |
| **Sh** | Shots that lead to another shot attempt |
| **Fld** | Fouls drawn that lead to a shot attempt |
| **Def** | Defensive actions that lead to a shot attempt |
| **PassLive.1** | Completed live-ball passes that lead to a goal |
| **PassDead.1** | Completed dead-ball passes that lead to a goal. |
| **TO.1** | Successful take-ons that lead to a goal |
| **Sh.1** | Shots that lead to another goal-scoring shot |
| **Fld.1** | Fouls drawn that lead to a goal |
| **Def.1** | Defensive actions that lead to a goal |
| **TotDist** | Total distance, in yards, that completed passes have traveled in any direction |
| **PrgDist** | Total distance, in yards, that completed passes have traveled towards the opponent's goal. |
| **Cmp.1** | Passes Completed between 5 and 15 yards |
| **Att.1** | Passes Attempted between 5 and 15 yards |
| **Cmp%.1** | Pass Completion Percentage between 5 and 15 yards |
| **Cmp.2** | Passes Completed between 15 and 30 yards |
| **Att.2** | Passes Attempted between 15 and 30 yards |
| **Cmp%.2** | Pass Completion Percentage between 15 and 30 yards |
| **Cmp.3** | Passes Completed greater than 30 yards |
| **Att.3** | Passes Attempted greater than 30 yards |
| **Cmp%.3** | Pass Completion Percentage greater than 30 yards |
| **xA** | The likelihood each completed pass becomes a goal assists given the pass type, phase of play, location and distance. |
| **KP** | Passes that directly lead to a shot (assisted shots) |
| **1/3** | Completed passes that enter the 1/3 of the pitch closest to the goal |
| **PPA** | Completed passes into the 18-yard box |
| **CrsPA** | Completed crosses into the 18-yard box |

**Data collection**

The data collected for this project came mainly from flat files (.csv) and web-scraping. I also used the FPL API to harvest the information about my user account history.

**Flat Files:**

As previously stated, the FPL historical data is not accessible via the API. There is a github repository with csv files that has recorded the API data from previous seasons. This repository is where I accessed the data for the FPL statistics. The repository consists of a folder for each season. Within those folders there are two important sections for each season.

The first is the gameweek data, which is the data recorded for each gameweek within the season. This data is stored in a folder (gws) that has a csv file for each of the gameweeks and a csv file which merges all the gameweeks into one file (merged\_gw.csv)

The second is the player id information (player\_idlist.csv). The player id information file records the player’s first name, second name and the unique id that they were assigned that season.

**Web-Scraping:**

To access the data from FBREF I had to web-scrape it from the website. I has to scrape the data from the website for each season individually. For each season on the website there is a table for all the players recorded in that season. The table in the website is not easy to access as the information from the table is commented out within the html. To select the table I had to remove the comment section of the html code.

.replace("<!--"," ").replace("-->", " ")

Once the comment was accessible I could access the hyperlinks in the table for each player in the league that season. I then changed the URL I was requesting for each relevant season I wanted to access.

Within those hyperlinks were the url\_ids and url\_names for each player. I extracted those parts from the hyperlinks using splitting and indexing. I then added them to a dataframe to store them. It was important to extract them as these were used in the individual player data URLs for each season.

Now that I had the required url\_id and tag for each player in the league I could obtain the three tables that I wanted for each player in each season. I had to loop through the dataframes I had stored and read the tables into new dataframes that I stored. I had to make sure I waited between each request otherwise I would be blocked by the website for 12 hours.

for x in range(0, (len(df)-1)):

    url1 = f"https://fbref.com/en/players/{df['url\_id'][x]}/matchlogs/{pl\_season}-{pl\_season + 1}/c9/summary/{df['tag'][x]}-Match-Logs"

    summary = pd.read\_html(url1)[0]

    summary = summary.dropna()

    summary.columns = [col[1] for col in summary.columns]

    summary.to\_csv(f"{pl\_season}\_stats/{df['id'][x]}\_summary.csv")

    sleep(3)

    url2 = f"https://fbref.com/en/players/{df['url\_id'][x]}/matchlogs/{pl\_season}-{pl\_season + 1}/c9/passing/{df['tag'][x]}-Match-Logs"

    passing = pd.read\_html(url2)[0]

    passing = passing.dropna()

    passing.columns = [col[1] for col in passing.columns]

    passing.to\_csv(f"{pl\_season}\_stats/{df['id'][x]}\_passing.csv")

    sleep(3)

    url3 = f"https://fbref.com/en/players/{df['url\_id'][x]}/matchlogs/{pl\_season}-{pl\_season + 1}/c9/gca/{df['tag'][x]}-Match-Logs"

    gca = pd.read\_html(url3)[0]

    gca = gca.dropna()

    gca.columns = [col[1] for col in gca.columns]

    gca.to\_csv(f"{pl\_season}\_stats/{df['id'][x]}\_gca.csv")

    print(x)

    sleep(3)

After this was completed, I had obtained all of the data I needed from FBREF I had 3 tables for every player in every season:

1. Summary Statistics
2. Passing Statistics
3. Goal Creating Action Statistics

These statistics were the ones that appeared to be the most required as they contained the ‘expected’ metrics and they were the most relevant to how points are awarded in FPL.

**API:**

To access the information from my team’s history in the 2022/23 season I used the FPL API. The base URL for the API is:

url = 'https://fantasy.premierleague.com/api/'

I then had to find the endpoint path to my account information which is:

**base\_URL + entry/{manager\_id}/history**

I then found my unique manager id through the website and could request information relevant to my account.

response = requests.get(f"{url}entry/{manager\_id}/history/")

The endpoint returned a dictionary which I could take the 2022/2023 season information from my account from.

info = response.json()

info['past'][1]

{'season\_name': '2022/23', 'total\_points': 2405, 'rank': 622876}

**Big Data System:**

I couldn’t find a public Big Data System that had data related to football or FPL so instead of harvesting data from online I will add my data to a Big Data System. I used Google BigQuery which I have access to from Ironhack.

I added both my final fbref table and fpl table into google cloud service:

from google.cloud import bigquery

fbref\_df.to\_gbq("ollie.fbref\_data", project\_id='da-bootcamp-2023', if\_exists='replace')

**Data cleaning/wrangling**

With all the dataframes I obtained I checked for duplicate rows and null values and couldn’t find any. For the Data Types for the columns in each dataframe I waited until I had merged all the data together and added it to the database before handling whether the column should be a Float, Integer, String or Date. I waited as I didn’t want there to be any mismatched columns before merging the seasons.

**Naming Problem;**

To combine the FPL to the FBREF data I had to make sure that the names of the players playing in each season were synced together so that I can access data from both of the dataframes and combine them for analysis and modelling.

An immediate problem I encountered after obtaining the player names and URL\_ids from FBREF was that the names stored in the tables were sometimes either nicknames or shortened names. I needed the full names to be able to sync the data to the FPL data. To access the full names, I went to the URL for each player and requested the name given in the table and the full name of the player.

fbref\_list = []

for x in range(0,len(df)):

    url\_name = f"https://fbref.com/en/players/{df.iloc[x]['url\_id']}/{df.iloc[x]['tag']}"

    reply = requests.get(url\_name)

    soup2 = BeautifulSoup(reply.content, 'html.parser')

    try:

        name1 = soup2.select('div:nth-child(2) > h1')[0].text.split('\n')[1]

    except:

        name1 = None

    try:

        name2 = soup2.select('div:nth-child(2) > p:nth-child(2) > strong')[0].text

    except:

        name2 = None

    if name2 == 'Position:':

        name2 = name1

    fbref\_list.append(name2)

    sleep(3)

I then had a list of all names from FBREF for each season and I also had the FPL names from the player\_idlist.csv file for each season. I stored all the names in a dataframe where every name in FBREF was merged with every name in FPL.

The problem was still not solved. The names recorded in FPL were not always the full name that was stored in the FBREF and vice-versa. To try and catch all the names I defined a function that checked for if the name from one source was contained in the other or the other way around.

def filter\_name(row):

    split1 = row.fbref\_name.split()

    split2 = row.full\_name.split()

    if all([x in row.full\_name for x in split1]) or all([x in row.fbref\_name for x in split2]):

        return True

    else:

        return False

I then iterated through the dataframe for all the names and returned a new dataframe which only kept the names if they matched. I realized that I had lost some names. The check was over constrained but if I loosened the check at all it created many duplicate rows that were too many to filter through individually. I also realized that because I had data from 6 seasons of football I would still have enough data to analyze and train a model from. The important factor was that I was only dropping names from FBREF so I still had the data from FPL intact and could use the expected statistics from that for the testing of the model.

**Syncing Data:**

The dataframe that stored the matching names could be used to look at data from both sources. However, I wanted to concatenate all the seasons together and the id provided by FPL was only unique to the season from which it was taken. I created a new unique\_id for each player in all datasets. The unique id was a combination of the FPL id and a specific integer for each season.

**Exploratory Data Analysis**

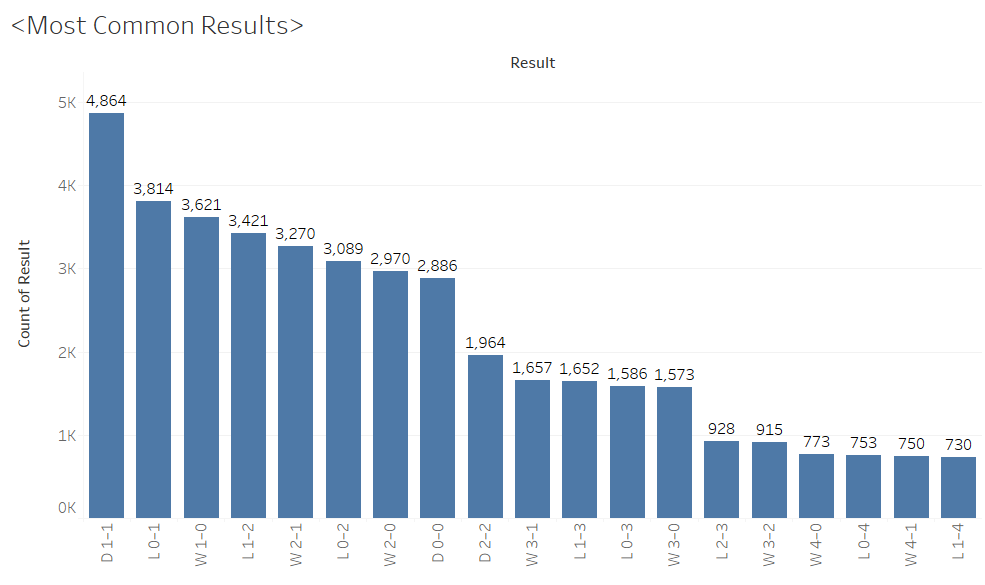
**Home v Away:**

A screenshot of a computer

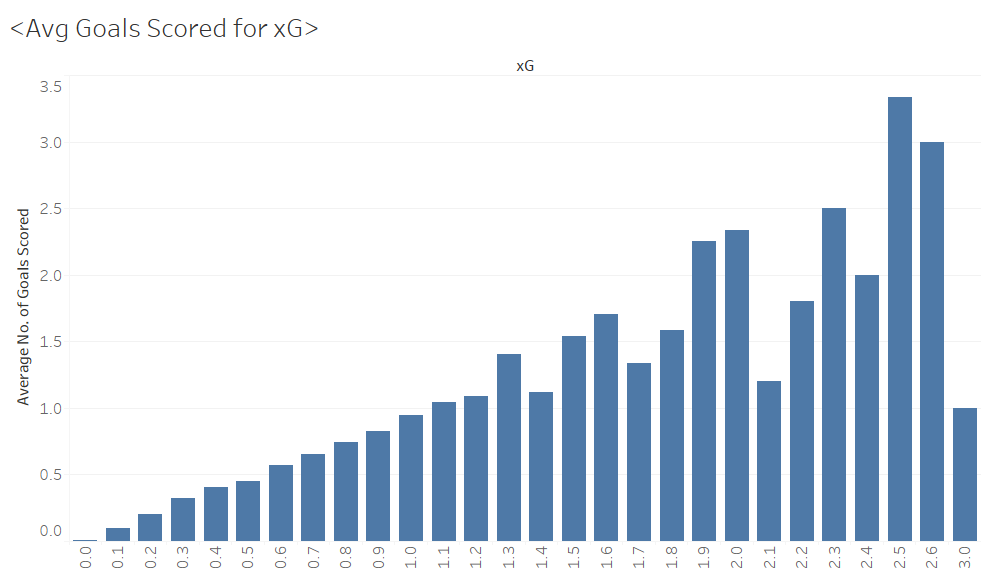
Description automatically generatedThe first thing to look at is whether playing at home or away is a factor. I’ll look at the top 7 performing clubs from the Premier League and measure their total goals scored for both home and away.

**Most Common Results:**

The next thing to look at is what are the most common results in the Premier League from all games where D = draw, L= home team lose, W = home team win.

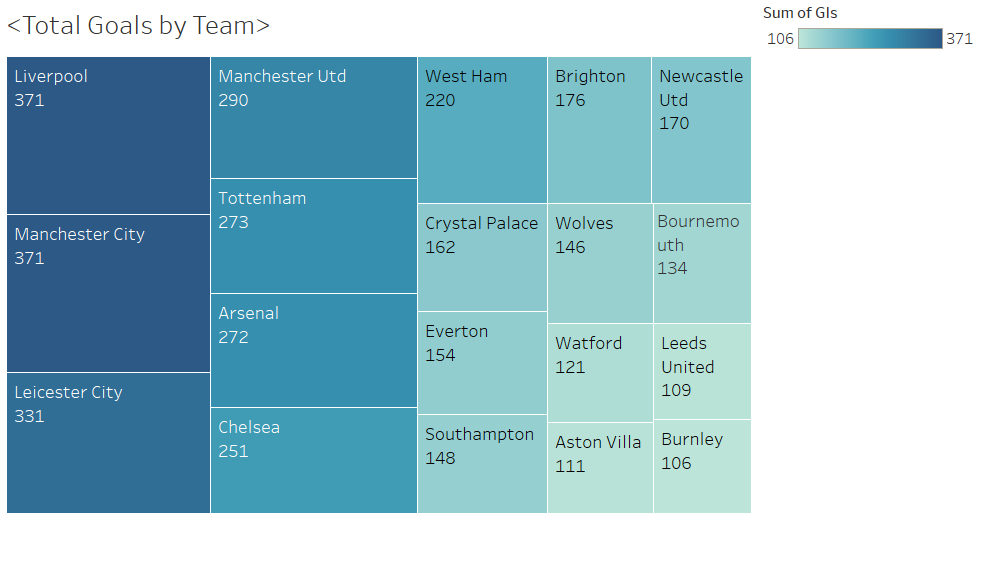
****

**Goals to Expected Goals (xG):**

**** The next thing to look at is whether the goals scored is correlated to expected goals.

**Goals Scored by teams:**

The next thing to look at is the which teams have scored the most goals for teams that have scored more than 100 goals over the 6 seasons.

****

**Correlation Matrix:**

The next thing to look at is a correlation matrix for the FPL data and what metrics are most correlated to the total points scored by the players each week.

A colorful squares with white text

Description automatically generated with medium confidence

**Database type selection**

There are two main types of database that are used. The differences between the databases are based on how they store and manage the data:

**Relational Database (SQL):**

Relational databases store data in schemas. The schemas consist of tables containing rows and columns. The tables each store different entities. The rows consist of an instance of the entity where the column stores the entity attributes. For any given row/column combination within a table a datapoint will exist.

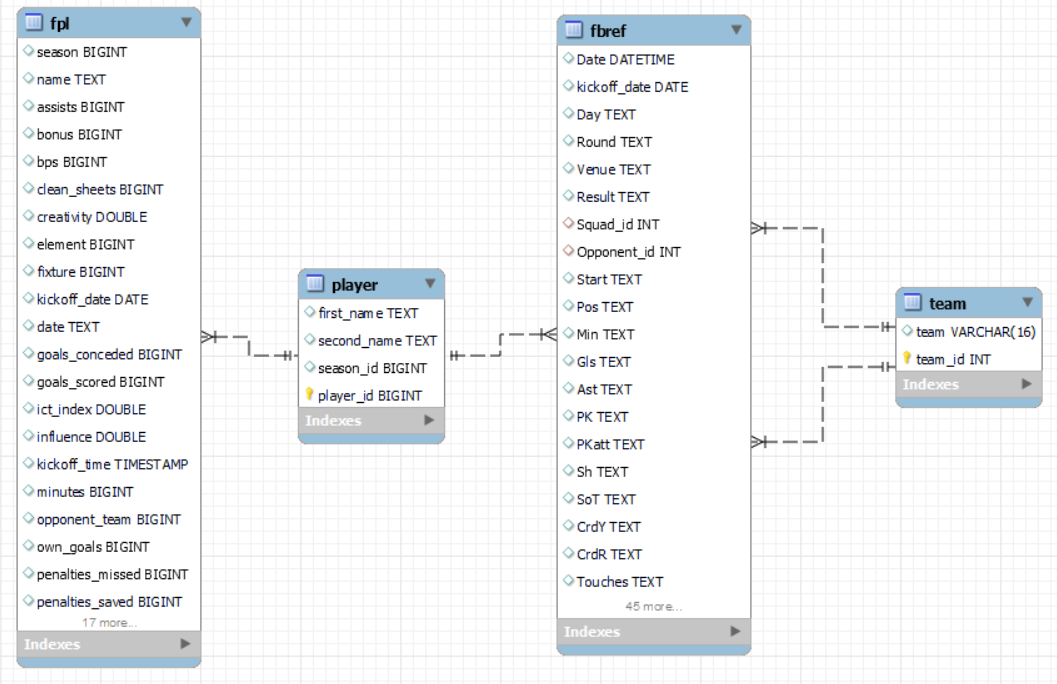
The tables in relational databases can be linked to each other through the use of keys. A primary key in one table will be exclusively unique values within that table and another table can link to it through a foreign key where the values do not have to be unique.

**Non-Relational Databases (NoSQL):**

Non-Relational databases do not require a predefined schema. They do not require structured data within tables so each entry in the database can have its own structure. They data can be a key-value, a graph a document etc.

The data I have is structured in tables and I need to link them through keys so a Relational Database seems to be the better option. It is also helpful for if I want to add more data as it is vertically scalable so it is easy to add a new season of data.

**MySQL Queries and ERD**

**ERD:**

**Primary Keys:**

player\_id in player

team\_id in team

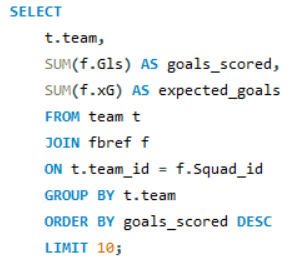
**Foreign Keys:**

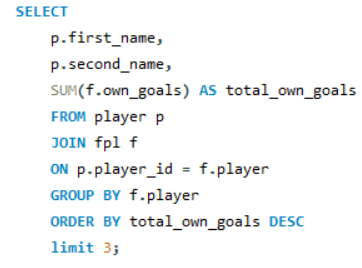
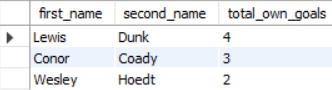
player in fpl – PK = (player\_id in player)

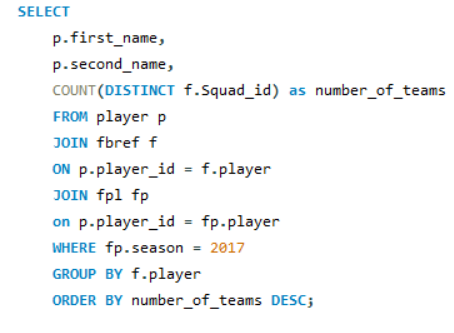
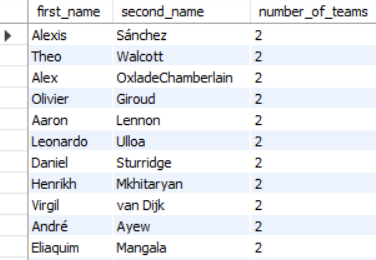
player in fbref – PK = (player\_id in player)

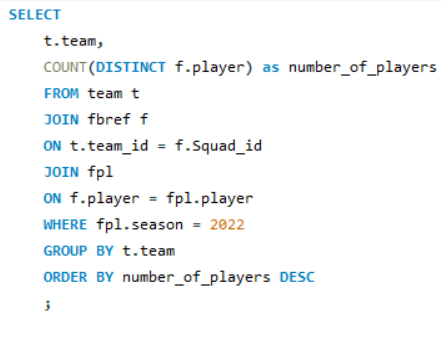
player in fbref – PK = (team\_id in team)

**MySQL Queries:**

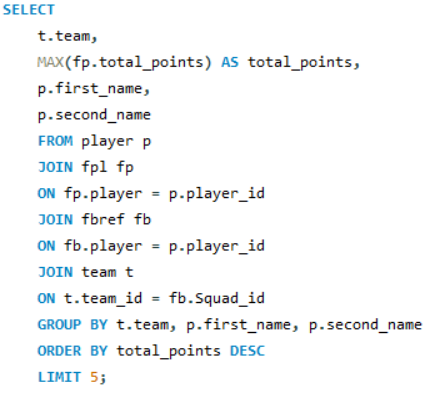
Show the top 10 goal scoring teams and their expected goals.

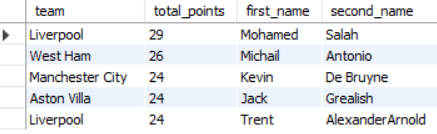
Show the top 3 players who scored the most own goals in a season.

Show the player who were transferred from one premier league club to another in the 2017 season.

Show how many players played for each club in the 2022 season.

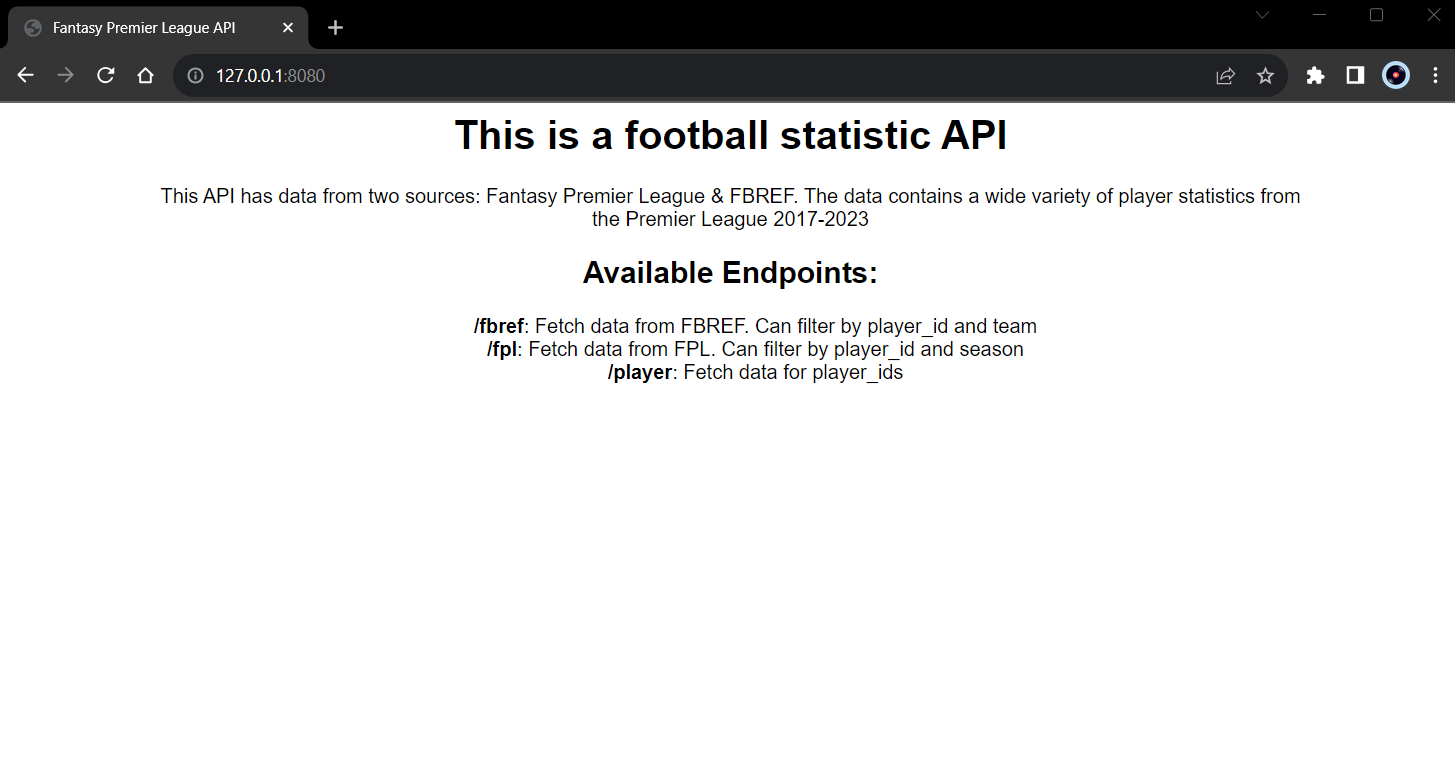
Show the highest scoring players and the clubs they played for in all seasons of fpl.





**Exposing Data via API**

I created an API to expose a part of the data. I created the API using flaskapi in a python file using HTML.

This is the homepage:

For the FBREF endpoint:

@app.route('/fbref', methods=['GET'])

def get\_big\_query\_comments():

    player = request.args.get('player')

    team = request.args.get('team')

    query = "SELECT p.first\_name, p.second\_name, f.player, t.team, f.kickoff\_date, f.Gls AS goals, f.xG AS expected\_goals FROM player p JOIN fbref f ON f.player = p.player\_id JOIN team t ON t.team\_id = f.Squad\_id "

    params = []

    if player:

        query += "AND f.player = %s "

        params.append(player)

    if team:

        query += "AND team = %s "

        params.append(team)

    conn = get\_db\_connection()

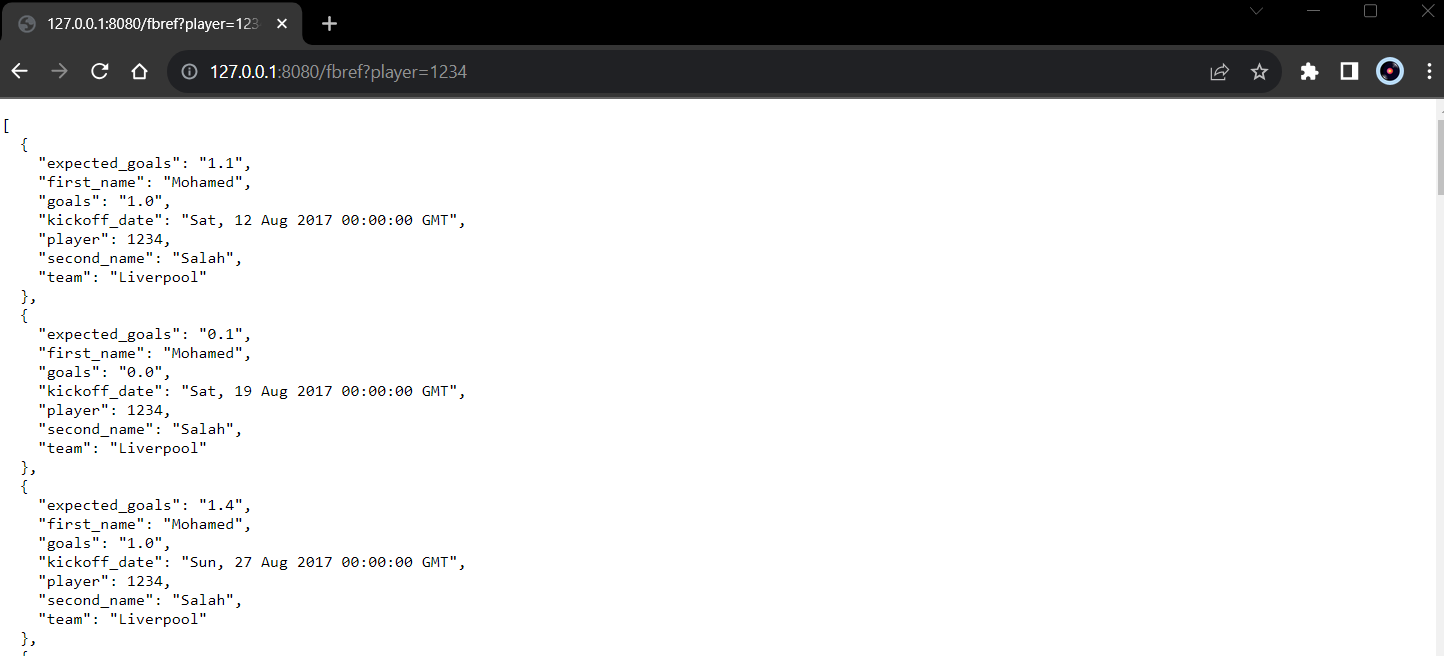
    cursor = conn.cursor()

    cursor.execute(query, tuple(params))

    comments = cursor.fetchall()

    conn.close()

    return jsonify(comments)

How the FBREF endpoint looks in browser: 

**Notes**

**Trello link =** [**https://trello.com/b/LuDj3hbH/fplproject**](https://trello.com/b/LuDj3hbH/fplproject)

**Github link =**

**To Do:**

Continue with further EDA with the aim to use the best metrics for machine learning

Create the machine learning model to improve on my FPL score from last season

Improve the flaskapi endpoints and usability