

Stage 1. Find and critique a Dataset

Asses the Dataset

With the chosen domain of Football, there were a number of datasets available to select from. These datasets contained different attributes, different scopes or different sizes. However, this particular dataset was very interesting to me as there were a variety of scoring metrics to select from as well as the name of team, name of league and year. In addition, many of the attributes in the dataset appealed to me because I find that scoring metrics are very important in real life football as seen from the amount of data analysts in the game nowadays.

Football is an established sport and some might even say that it's the best sport in the world due to it's entertainment value. I believe that the key driving force of it's entertainment value is the fact that teams nowadays place a huge importance in the analysis of their performances through scoring metrics. Hence, I decided to have my take in exploring this dataset. Using this dataset together with assumptions or proven statistics, such as 'goals wins games', we would ask some questions.

One question I would like to query is "Which team has scored the most goals?". As mentioned earlier in the report, usually one would have the idea of thinking that goals would win a game of football, however, are there any factors that assist this claim? That is why I'll be moving on to my second question, "Is there a correlation between goals scored and amount of games won?", this allows us to find out if the attribute of goals scored affects the amount of games won. The last question I would like to ask is "Is there a correlation between xG, xpts and the performance of the team from Q1", this allows us to find out if there are any other factors that affect the performances of the team.

Q1: Which team has scored the most goals?

Q2: Is there a correlation between goals scored and amount of games won?

Q3: Is there a correlation between xG, xpts and the performance of the team?

Terms of Use

The dataset that is being utilized contained clean data where different teams from different years and leagues had different attributes such as xG, xG_diff and scored. This information was found on a website called Kaggle.

In terms of licensing, it just says that this particular Notebook has been released under the Apache 2.0 open source license and on the actual website where the dataset was from, understat.com, it doesn't say anything about licensing except for a copyright. This means that the data is free to be used by any individual with the restriction of a copyright protection to cite understat as the source of the data.

Hence, we will cite the source from understat.

Quality

The source that I've gotten the data from is a website named Kaggle where various types of datasets can be obtained from. Kaggle provides data for free and available to the public. The author of this dataset is understat, a website with football data for 6 european leagues for every season since 2014/15 season. (Understat API, n.d.)

This website has been widely used all over Kaggle so there is some form of quality in the dataset, if not the users on Kaggle wouldn't want to use the dataset.

Level of Detail

The dataset has a high level of detail as it provides various scoring metrics which will help in the deriving of answers for the questions. Having a range of scoring metrics to choose from, allows me to find out which scoring metric affect the amount of games won by a specific team.

Documentation

As an avid football fan, there wasn't much difficulty faced when trying to interpret the data as the attributes were labelled clearly and I could understand them due to my football knowledge. However, this may be different for a general user that doesn't watch football or take an interest in football statistics as they wouldn't be able to understand "deep_allowed", "xG"

and "ppda_coef". With that being said, on Kaggle, the description of each attribute was specified, so it gives a basic understanding of each attribute but I feel like the general user should do more research on certain attributes.

Interrelation

The dataset could be combined with other datasets but I believe that it will be very complicated as it would require the other datasets to be of the same year, league and team name. Once it fulfills those requirements then only can you add on other attributes to the dataset.

Use

In this project, the database is being utilized to perform hypothesis testing where it validates if a specific attribute such as xG affects the performance of a specific team. I believe that the dataset has the potential to be utilized in a prediction machine learning model. But since the scope of this assignment is to query from the database, the prediction machine learning model will not be implemented.

Discoverability

To sum up, the dataset was relatively simple to open up in my selected domain of Football as there were several users on Kaggle that provided me with differing scopes of datasets related to the domain. But I feel that other datasets were not as specific and interesting as this scope.

Stage 2. Model your Data

The figure below (Fig 1) shows the ER model of the data where the Leagues table has a one to many relationship with the Teams table, linked by a foreign key. To give an example, one league can have many teams which is linked by the team_leagues. The Team table and Stats table has a one to many relationship as well, where one team can have many statistics, linked by the

team. The Season table has a one to many relationship with the Stats table, as one season can have many statistics, linked by the year.

These relationships will help to maintain data consistency and integrity across the different database tables, allowing for a more meaningful querying experience and analysis.

Figure 2 (Fig 2) is a visual representation of how the tables connect with one another, using foreign keys.

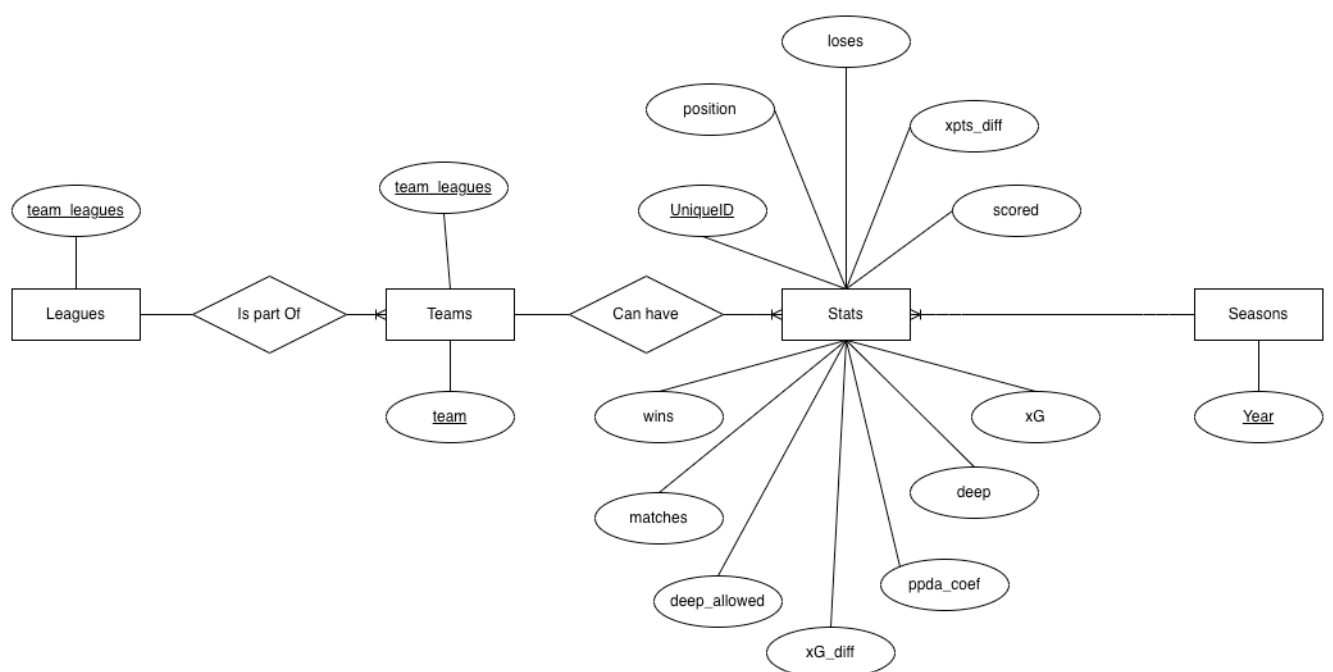


Fig 1. ER Model

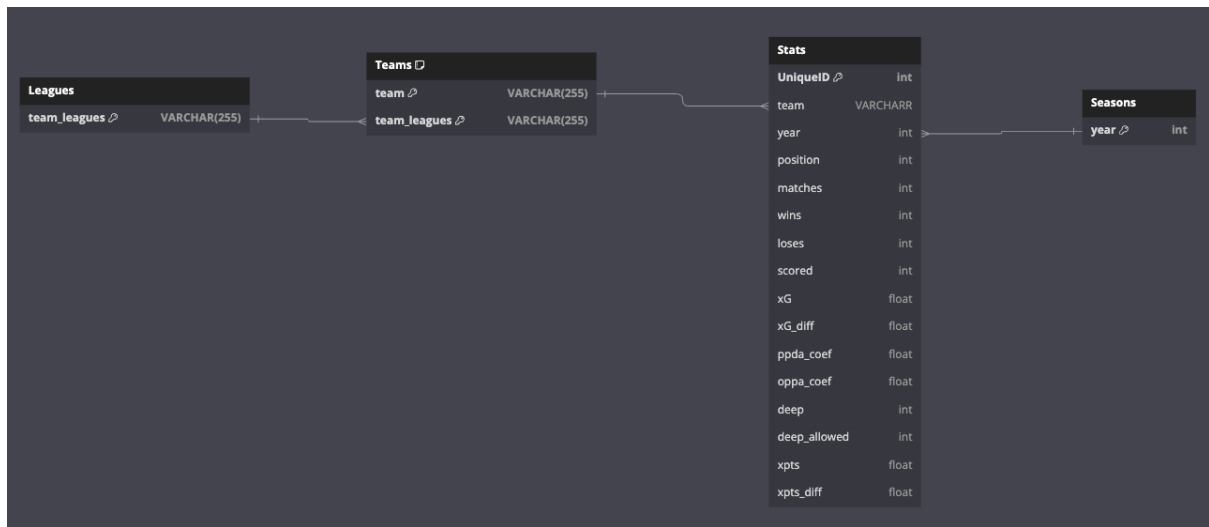


Fig 2. Database Schema

The table below basically shows all the attributes in the dataset that I've chosen to utilize as I believe that other attributes will not contribute to the correlation of performance.

Attributes	Description
Team_leagues	Name of the league
Year	The value of year
Position	Position of league
Team	Name of team
Matches	Amount of matches played
Wins	Amount of games won
Loses	Amount of games lost
Scored	Amount of goals scored
xG	A scoring metric to determine the probability of a shot resulting in a goal
xG_diff	Difference between actual goals scored and expected goals.
Ppda_coef	Amount of passes a team is allowed to make before the opponent makes a defensive action
Oppda_coef	Amount of passes the opponent is allowed to make before the team makes a defensive action

Deep	Amount of deep passes made
Deep_allowed	Amount of deep passes conceded
Xpts	Expected points
Xpts_diff	Difference in the expected points
UniqueID	A unique ID to identify the rows

Normalisation

All of the tables are normalized in Boyce-Codd Normal Form (BCNF) as for the Leagues and Seasons table they do not have any other attributes other than their primary keys. For the Stats table it is in BCNF as all non-prime attributes are fully functionally dependent on primary keys of (UniqueID, Team, Year). The Teams table is in BCNF as every non-trivial functional dependency is a private key.

Stage 3. Create the Database

Record all CREATE commands

The create commands was made in google colab and imported into the coursera terminal through the create-tables.sql.

In google colab, a database called Football_Stats was created:

```
DROP DATABASE IF EXISTS Football_Stats;
CREATE DATABASE Football_Stats;
```

Secondly, a database user called 'marcus' was created to grant access to the database 'Football_Stats' with the password of 'password123':

```
DROP USER IF EXISTS 'marcus'@'%';
CREATE USER 'marcus'@%' IDENTIFIED WITH mysql_native_password BY
'password123';
```

Thirdly, I needed to grant the user 'marcus' access to the project data, along with server side configurations

```
GRANT ALL ON Football_Stats.* TO 'marcus'@'%';
GRANT SELECT ON mysql.* TO 'marcus'@'%';
```

Lastly, I created the main table to store the raw data from the csv file:

```
%%writefile $SCRIPT_PATH/create-tables.sql
```

```
USE Football_Stats;
```

```
CREATE TABLE Leagues (  
    team_leagues VARCHAR(255) PRIMARY KEY  
);
```

```
CREATE TABLE Teams (  
    team VARCHAR(255) PRIMARY KEY,  
    team_leagues VARCHAR(255),  
    FOREIGN KEY (team_leagues) REFERENCES Leagues(team_leagues)  
);
```

```
CREATE TABLE Seasons (  
    year INT PRIMARY KEY  
);
```

```
CREATE TABLE Stats (  
    UniqueID INT PRIMARY KEY,  
    team VARCHAR(255),  
    year INT,  
    position INT,  
    matches INT,  
    wins INT,  
    loses INT,  
    scored INT,  
    xG FLOAT,  
    xG_diff FLOAT,  
    ppda_coef FLOAT,  
    oppda_coef FLOAT,
```

```
deep INT,  
deep_allowed INT,  
xpts FLOAT,  
xpts_diff FLOAT,  
FOREIGN KEY (team) REFERENCES Teams(team),  
FOREIGN KEY (year) REFERENCES Seasons(year)  
);
```

This saves the tables into the create-tables.sql file:

```
!mysql -t < /home/coder/project/suicides/scripts/create-tables.sql
```

Enter instance data

This is where I combined all of the tables into a denormalised table to write it to the load-denorm-data.sql. The point of creating a denormalised table is to temporarily store the denormalised data and pivot it into a tall table.

```
%%writefile $SCRIPT_PATH/load-dnorm-data.sql
```

```
USE Football_Stats;
```

```
DROP TABLE IF EXISTS denormalised;
```

```
CREATE TABLE denormalised (  
  teamLeagues VARCHAR(255),  
  year INT(4),  
  position INT(2),  
  teamNames VARCHAR(255),  
  matches INT(2),  
  wins INT(2),  
  loses INT(2),  
  scored INT(3),  
  xG FLOAT,  
  xG_diff FLOAT,
```



```
ppdaCoef FLOAT,  
oppdaCoef FLOAT,  
deep INT(3),  
deepAllowed INT(3),  
xpts FLOAT,  
xptsDiff FLOAT,  
unique_id INT(11)  
);
```

LOAD DATA INFILE

'/home/coder/project/Football_Stats/data/understat.com.csv'

INTO TABLE denormalised

FIELDS TERMINATED BY ','

ENCLOSED BY ''

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

*(teamLeagues, year, position, teamNames, matches, wins, loses, scored, xG,
xG_diff, ppdaCoef, oppdaCoef, deep, deepAllowed, xpts, xptsDiff, unique_id);*

This then saves the denormalised table into the load-dnorm-data.sql:

```
!mysql -t < /home/coder/project/Football_Stats/scripts/load-dnorm-data.sql'
```

I then added the denormalised data into the main tables using the following:

```
%%writefile $SCRIPT_PATH/ingest-data.sql
```

```
USE Football_Stats;
```

```
DELETE FROM Stats;
```

```
DELETE FROM Seasons;
```

```
DELETE FROM Teams;
```

```
DELETE FROM Leagues;
```

```
INSERT INTO Leagues (team_leagues)
SELECT DISTINCT teamLeagues
FROM denormalised WHERE teamLeagues IS NOT NULL;
```

```
INSERT INTO Teams (team)
SELECT DISTINCT teamNames
FROM denormalised WHERE teamLeagues IS NOT NULL;
```

```
INSERT INTO Seasons (year)
SELECT DISTINCT year
FROM denormalised WHERE teamLeagues IS NOT NULL;
```

```
INSERT INTO Stats (year, team, position, matches, wins, loses, scored, xG,
xG_diff, ppda_coef, oppda_coef, deep, deep_allowed, xpts, xpts_diff,
UniqueID)
SELECT DISTINCT year, teamNames, position, matches, wins, loses,
scored, xG, xG_diff, ppdaCoef, oppdaCoef, deep, deepAllowed, xpts, xptsDiff,
unique_id
FROM denormalised WHERE teamLeagues IS NOT NULL;
```

Next I saved the table into the ingest-data.sql using:

```
!cp $SCRIPT_PATH/*.sql $GD_SCRIPT_PATH/.
```

```
!cp $DATA_PATH/*.csv $GD_DATA_PATH/.
```

Reflection on how well the database reflects the data

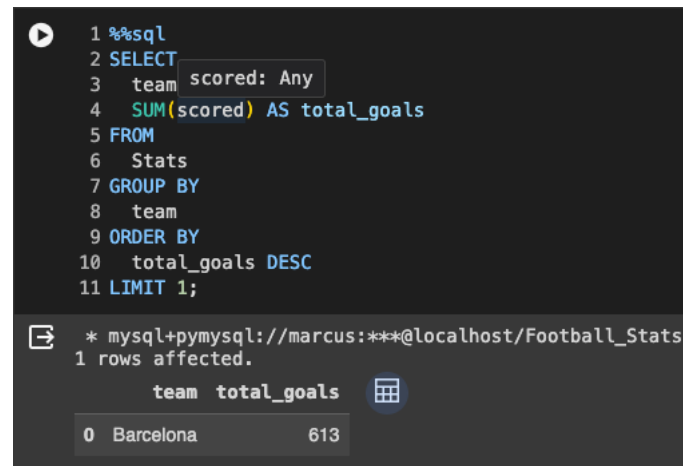
The database accurately reflects the data with all values filled in.

List SQL commands that answer questions

Q1: Which team has scored the most goals?

The row of the table displays the team with the highest total goals scored across all seasons along with the amount of goals. As for the columns it displays the attributes of the table, team and total_goals. For this I utilized the

mathematical function, sum, to calculate the total_goals. 'Limit' to show only the top result. This will be used as the team for the latter questions.



```
1 %%sql
2 SELECT
3   team
4   SUM(scored) AS total_goals
5 FROM
6   Stats
7 GROUP BY
8   team
9 ORDER BY
10  total_goals DESC
11 LIMIT 1;
```

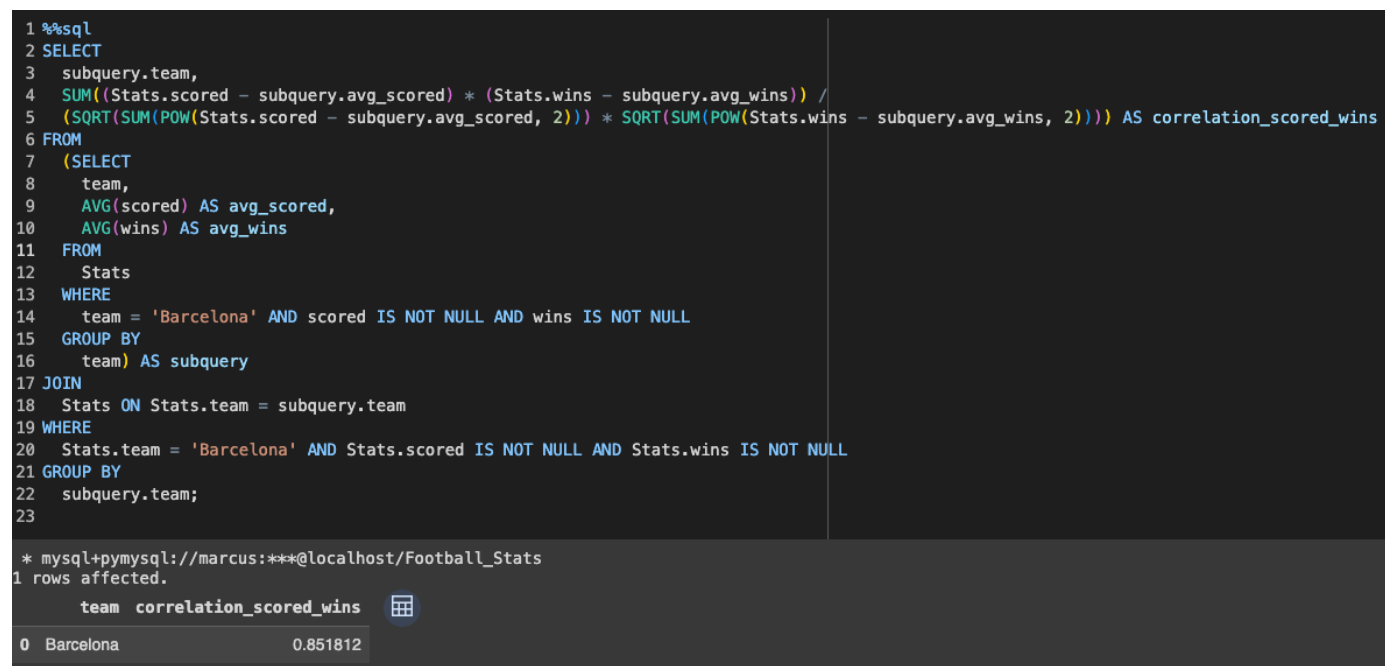
* mysql+pymysql://marcus:***@localhost/Football_Stats
1 rows affected.

	team	total_goals
0	Barcelona	613

Fig 3. SQL statement of Q1 and Results

Q2: Is there a correlation between goals scored and amount of games won?

The sql statement for Q2 utilizes a variety of mathematical functions to calculate correlation values, such as SUM, SQRT, AVG and POW. This is then matched against the average value of goals scored and wins from the stats table for the team Barcelona through the use of the JOIN statement. The result show that there is a relatively strong correlation between goals scored and games won as it has a score of 0.85 (2d.p). (Fig 4)



```
1 %%sql
2 SELECT
3   subquery.team,
4   SUM((Stats.scored - subquery.avg_scored) * (Stats.wins - subquery.avg_wins)) /
5   (SQRT(SUM(POW(Stats.scored - subquery.avg_scored, 2))) * SQRT(SUM(POW(Stats.wins - subquery.avg_wins, 2)))) AS correlation_scored_wins
6 FROM
7   (SELECT
8     team,
9     AVG(scored) AS avg_scored,
10    AVG(wins) AS avg_wins
11  FROM
12    Stats
13  WHERE
14    team = 'Barcelona' AND scored IS NOT NULL AND wins IS NOT NULL
15  GROUP BY
16    team) AS subquery
17 JOIN
18   Stats ON Stats.team = subquery.team
19 WHERE
20   Stats.team = 'Barcelona' AND Stats.scored IS NOT NULL AND Stats.wins IS NOT NULL
21 GROUP BY
22   subquery.team;
```

* mysql+pymysql://marcus:***@localhost/Football_Stats
1 rows affected.

	team	correlation_scored_wins
0	Barcelona	0.851812

Fig 4. SQL statement of Q2 and Results

Q3: Is there a correlation between xG, xpts and the performance of the team?

In this part I will investigate other attributes to see if they affect the performance of the team.

The sql statement uses the same formula as in Q2 to calculate correlation values. As seen from the results, xpts and xG have a stronger correlation score of 0.92 and 0.90 respectively, this means that goals scored isn't the only factor for winning games, and in fact there are stronger attributes. (Fig 5)

%%sql

SELECT

'corr_xG_wins' AS correlation_type,
*(SUM(xG * wins) - COUNT(xG) * AVG(xG) * AVG(wins)) / (SQRT((SUM(xG * xG) -*
*COUNT(xG) * AVG(xG) * AVG(xG)) * (SUM(wins * wins) - COUNT(wins) * AVG(wins) **
AVG(wins)))) AS correlation_value

FROM Stats

WHERE xG IS NOT NULL AND wins IS NOT NULL AND team = 'Barcelona'

UNION

SELECT

'corr_xG_losses' AS correlation_type,
*(SUM(xG * losses) - COUNT(xG) * AVG(xG) * AVG(losses)) / (SQRT((SUM(xG * xG) -*
*COUNT(xG) * AVG(xG) * AVG(xG)) * (SUM(losses * losses) - COUNT(losses) * AVG(losses) **
AVG(losses)))) AS correlation_value

FROM Stats

WHERE xG IS NOT NULL AND losses IS NOT NULL AND team = 'Barcelona'

UNION

SELECT

'corr_xpts_wins' AS correlation_type,
*(SUM(xpts * wins) - COUNT(xpts) * AVG(xpts) * AVG(wins)) / (SQRT((SUM(xpts * xpts) -*
*COUNT(xpts) * AVG(xpts) * AVG(xpts)) * (SUM(wins * wins) - COUNT(wins) * AVG(wins) **
AVG(wins)))) AS correlation_value

FROM Stats

WHERE xpts IS NOT NULL AND wins IS NOT NULL AND team = 'Barcelona'

UNION

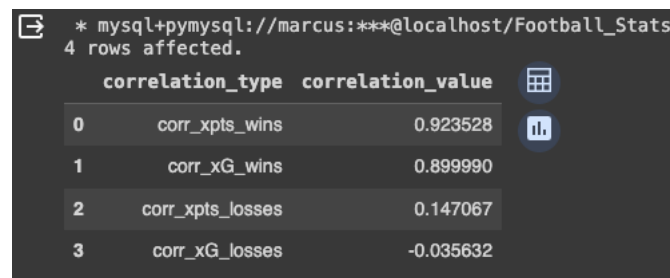
SELECT

'corr_xpts_losses' AS correlation_type,
$$\frac{(SUM(xpts * losses) - COUNT(xpts) * AVG(xpts) * AVG(losses))}{(SQRT((SUM(xpts * xpts) - COUNT(xpts) * AVG(xpts) * AVG(xpts)) * (SUM(losses * losses) - COUNT(losses) * AVG(losses) * AVG(losses))))}$$
 AS correlation_value

FROM Stats

WHERE xpts IS NOT NULL AND losses IS NOT NULL AND team = 'Barcelona'

ORDER BY correlation_value DESC;



	correlation_type	correlation_value
0	corr_xpts_wins	0.923528
1	corr_xG_wins	0.899990
2	corr_xpts_losses	0.147067
3	corr_xG_losses	-0.035632

Fig. 5 Results of Q3

Stage 4. Create a simple web application

Before creating the web application, I had to import the sql scripts mention earlier in the report because I did all of the sql scripts on google collab. (Fig 6)

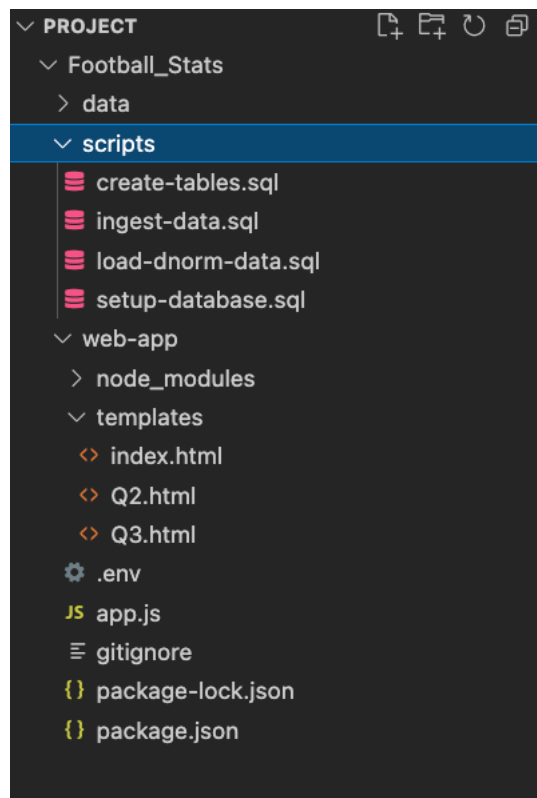


Fig. 6 Importing of sql scripts

Next I started to create the app.js where it contains all of the routing and sql statements. The user has to enter node app.js to run the website in order to create a connection, the password and user name is in the .env file. (Fig 6) The env file is then linked to the app.js through the constant 'env'. (Fig 7) This allows the user to gain access to the database and allows the system to query the database.

```

Football_Stats > web-app > JS app.js > app.get('/index') callback
1  const express = require('express');
2  const bodyParser = require('body-parser');
3  const mysql = require('mysql');
4  const mustacheExpress = require('mustache-express');
5  const env = require('dotenv').config();
6
7  const app = express();
8  const port = 3000;
9
10 app.engine('html', mustacheExpress());
11 app.set('view engine', 'html');
12 app.set('views', './templates');
13 app.use(bodyParser.urlencoded({ extended: true }));
14
15 var dbcon = mysql.createConnection({
16   host: env.parsed.HOST,
17   user: env.parsed.USER_NAME,
18   password: env.parsed.PASSWORD,
19   database: env.parsed.DATABASE
20 });
21
22 dbcon.connect((err) => {
23   if (err) {
24     console.error('There is an error connecting to the database', err);
25     process.exit(1); // Exit the application if unable to connect
26   }
27   console.log('You have successfully connected to the database!');
28 });
29
30 function templateRenderer(template, res) {
31   return function (error, results, fields) {
32     if (error)
33       throw error;
34     res.render(template, { data: results });
35   }
36 }
37

```

Fig. 7 Connection app.js

In Fig 8, the codes below will query the database according to the sql statements specified and display the data when the web app is loaded successfully.

```

app.get('/index', function (req, res) {
  dbcon.query("SELECT team, SUM(scored) AS total_goals FROM Stats GROUP BY team ORDER BY total_goals DESC LIMIT 1;", templateRenderer('index', res));
})

app.get('/02', function (req, res) {
  dbcon.query("SELECT subquery.team, SUM((Stats.scored - subquery.avg_scored) * (Stats.wins - subquery.avg_wins)) / (SQRT(SUM(POW(Stats.scored - subquery.avg_scored, 2))) * SQRT(SUM(POW(Stats.wins - subquery.avg_wins, 2)))) AS correlation_type, (SUM(xG * wins) - COUNT(xG) * AVG(xG) * AVG(wins)) / (SQRT((SUM(xG * xG) - COUNT(xG) * AVG(xG) * AVG(xG)) * (SUM(wins * wins) - COUNT(wins) * AVG(wins) * AVG(wins)))) AS corr_xG_wins FROM Stats GROUP BY team ORDER BY correlation_type DESC LIMIT 1;", templateRenderer('02', res));
})

app.get('/03', function (req, res) {
  dbcon.query("SELECT team, SUM((Stats.scored - subquery.avg_scored) * (Stats.wins - subquery.avg_wins)) / (SQRT(SUM(POW(Stats.scored - subquery.avg_scored, 2))) * SQRT(SUM(POW(Stats.wins - subquery.avg_wins, 2)))) AS correlation_type, (SUM(xG * wins) - COUNT(xG) * AVG(xG) * AVG(wins)) / (SQRT((SUM(xG * xG) - COUNT(xG) * AVG(xG) * AVG(xG)) * (SUM(wins * wins) - COUNT(wins) * AVG(wins) * AVG(wins)))) AS corr_xG_wins FROM Stats GROUP BY team ORDER BY correlation_type DESC LIMIT 1;", templateRenderer('03', res));
})

```

Fig. 8 Routing of Pages & Querying of Database

The display of the Q1's data is done through the index.html where it will display the team and total goals attribute. (Fig. 9)

```
Football_Stats > web-app > templates > index.html > html > body > br
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="utf-8" />
6      <title>Which team has scored the most goals?</title>
7  </head>
8
9  <body>
10     <a href="/Q2">Q2</a>
11     <a href="/Q3">Q3</a>
12     <br>
13     <a>Q1: Which team has scored the most goals?</a>
14     <table style="border-collapse: collapse; width: 100%;">
15     <thead>
16     <tr>
17         <th>Team</th>
18         <th>Most Total Goals</th>
19     </tr>
20     </thead>
21     <tbody>
22         {{#data}}
23         <tr>
24             <td>{{team}}</td>
25             <td>{{total_goals}}</td>
26         </tr>
27         {{/data}}
28     </tbody>
29     </table>
30 </body>
31
32 <style>
33     table {
34         border-collapse: collapse;
35         width: 100%;
36     }
37
38     th,
39     td {
40         border: 1px solid #ddd;
41         padding: 8px;
42     }
43 </style>
44
45 </html>
```

Fig.9 index.html

In Fig. 10, the web app will display the index.html to show what was queried in the app.get functions.

localhost:3000/index

[Q2](#) [Q3](#)

Q1: Which team has scored the most goals?

Team	Most Total Goals
Barcelona	613

Fig. 10 Display of index.html

The display of the Q2's data is done through the Q2.html where it will display the team and correlation_scored_wins attribute. (Fig. 11)

```

Football_Stats > web-app > templates > Q2.html > html > body > br
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="utf-8" />
6      <title>Question 2</title>
7  </head>
8
9  <body>
10     <a href="/index">Q1</a>
11     <a href="/Q3">Q3</a>
12     <br>
13
14     <a>Q2: Is there a correlations between goals scored and wins for the team, Barcelona.</a>
15     <table style="border-collapse: collapse; width: 100%;">
16         <thead>
17             <tr>
18                 <th>Team</th>
19                 <th>Correlation Scored Wins</th>
20             </tr>
21         </thead>
22         <tbody>
23             <tr>
24                 <td>{{team}}</td>
25                 <td>{{correlation_scored_wins}}</td>
26             </tr>
27         </tbody>
28     </table>
29
30 </body>
31
32
33 <style>
34     table {
35         border-collapse: collapse;
36         width: 100%;
37     }
38
39     th, td {
40         border: 1px solid #ddd;
41         padding: 8px;
42     }
43 </style>
44 </html>
45

```

Fig. 11 Q2.html

In Fig. 12, the web app will display the Q2.html to show what was queried in the respective app.get functions.



Team	Correlation Scored Wins
Barcelona	0.8518117267383599

Fig. 12 Display of Q2

The display of the Q3's data is done through the Q3.html where it will display the correlation_type and correlation_value attribute of the team named 'Barcelona'. (Fig. 13)

```
Football_Stats > web-app > templates > Q3.html > html > body > br
4 <head>
5   <meta charset="utf-8" />
6   <title>Q3</title>
7 </head>
8
9 <body>
10   <a href="/index">Q1</a>
11   <a href="/Q2">Q2</a>
12   <br>
13   <a href="/Q3">Q3: Is there a correlation between xG, xpts and actual performance for the team, Barcelona?</a>
14   <table style="border-collapse: collapse; width: 100%;">
15     <thead>
16       <tr>
17         <th>Correlation Type</th>
18         <th>Correlation Value</th>
19       </tr>
20     </thead>
21     <tbody>
22       <tr>
23         <td>{{correlation_type}}</td>
24         <td>{{correlation_value}}</td>
25       </tr>
26     </tbody>
27   </table>
28
29 </body>
30
31
32 <style>
33   table {
34     border-collapse: collapse;
35     width: 100%;
36   }
37
38   th,
39   td {
40     border: 1px solid #ddd;
41     padding: 8px;
42   }
43 </style>
44
45 </html>
```

Fig. 13 Q3. html

In Fig. 14, the web app will display the Q3.html to show what was queried in the respective app.get functions.

[Q1 Q2](#)

Q3: Is there a correlation between xG, xpts and actual performance for the team, Barcelona?

Correlation Type	Correlation Value
corr_xpts_wins	0.9235281224375883
corr_xG_wins	0.8999899044007332
corr_xpts_losses	0.14706748262960695
corr_xG_losses	-0.035632221287629384

Fig. 14 Display of Q3

References:

Aknkaradeniz (2022) Football_ab_test, Kaggle. Kaggle. Available at:

<https://www.kaggle.com/code/aknkaradeniz/football-ab-test/data>

(Accessed: January 8, 2024).

Understat API. (n.d.). Available at:

<https://pypi.org/project/understatapi/#:~:text=Understat%20is%20a%20website%20with,and%20the%20Russian%20Premier%20League.>

(Accessed: January 8, 2024)