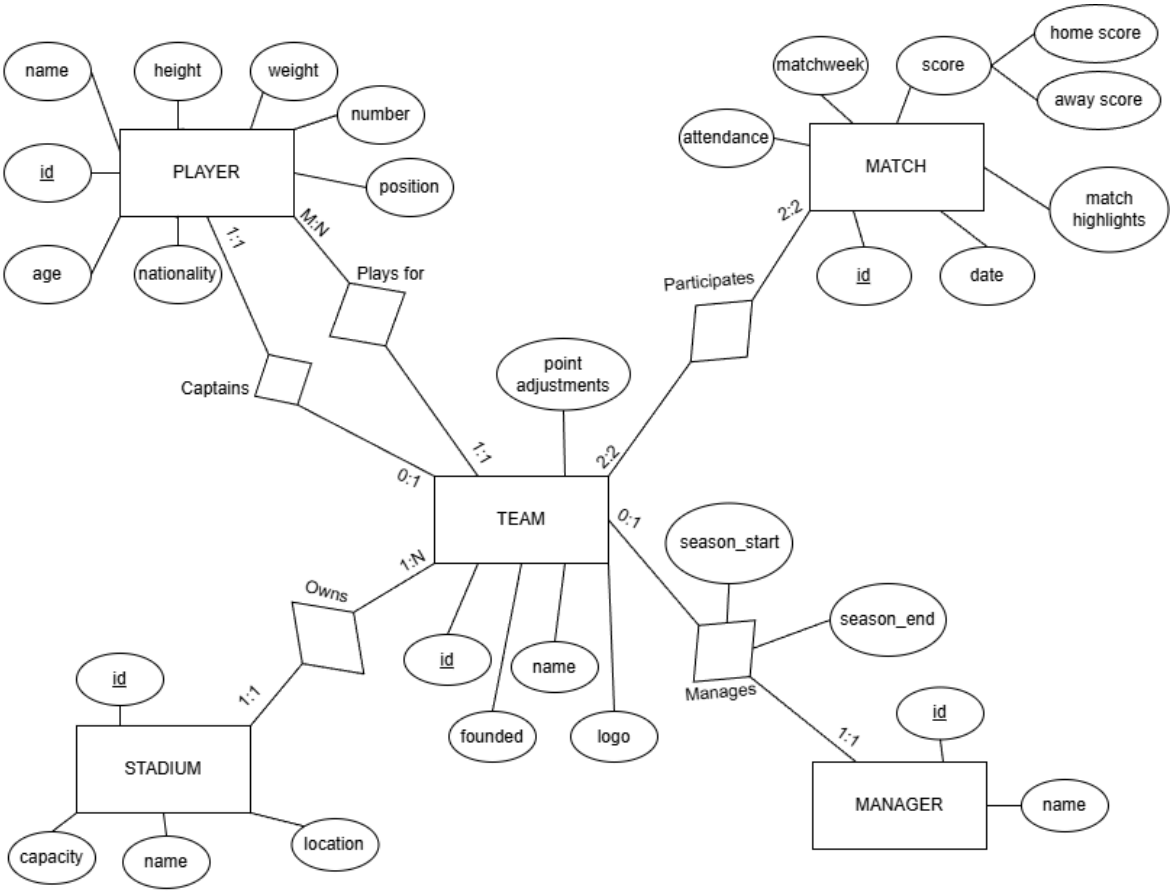
FP4 Report:
The Premier League Analytics Hub

By Nam Nguyen and Harrison Nguyen

1. ER Model and Relational Notation

a. ER Model



MODEL LIMITATIONS:

+ Players can switch teams in the middle of the season, which can affect the cardinality between Team and Player entities regarding the *plays_for* and *captains* relationship.

+ Players' jersey numbers should also be unique among players of the same club.

+Managers can switch teams in the middle of the season. At any given point in time, the relationship between the Manager and Team entities is 1:1. However, over time, the relationship becomes M:N, which is the chosen cardinality for the model.

+ Attendance should not be more than the capacity of a particular stadium.

+ Manager cannot end the season on a date that is before the *season_start* date.

+ Each team must participate in one matchweek no later than matchweek 18 and another matchweek starting from matchweek 19. (first and second legs).

b. Relational Notation

Team(id, team_name, founded_year, *stadium_id*, *captain_id*, logo_url)

captain_id: foreign key refers to id in Player, NULL not allowed, UNIQUE.

stadium_id: foreign key refers to id in Stadium, NULL not allowed.

Manager(id, name)

Managing(*manager_id*, *team_id*, season_start, season_end)

manager_id: foreign key refers to id in Manager, NULL not allowed.

team_id: foreign key refers to id in Team, NULL not allowed.

Player(id, name, number, age, nationality, position, weight, height)

Stadium(id, name, capacity, location)

Matches(id, date, *home_team_id*, *away_team_id*, home_score, away_score, matchweek, attendance, youtube_id)

home_team_id: foreign key refers to id in Team, NULL not allowed

away_team_id: foreign key refers to id in Team, NULL not allowed

Point_Adjustments(*team_id*, matchweek, adjustment, reason, date_applied)

team_id: foreign key refers to id in Team, NULL not allowed

---

2. How to Run the Program

GitHub: https://github.com/hmhngx/EPL-Database-Webiste

The DB: To build the database, first access GitHub and locate the data folder. Run the Postgressql_ddl.txt to build the layout of the database. Then, import the .CSV files into each and every corresponding table we just built.

The Data: Navigate to /etl, run pip install -r requirements.txt, and then python etl_script.py. This script pulls the "dirty" data, cleans it, and pushes it to Supabase.

The App: In the root folder, run npm install. Open two terminals:

Terminal 1: npm run server (The Express API)

Terminal 2: npm run dev (The React UI)

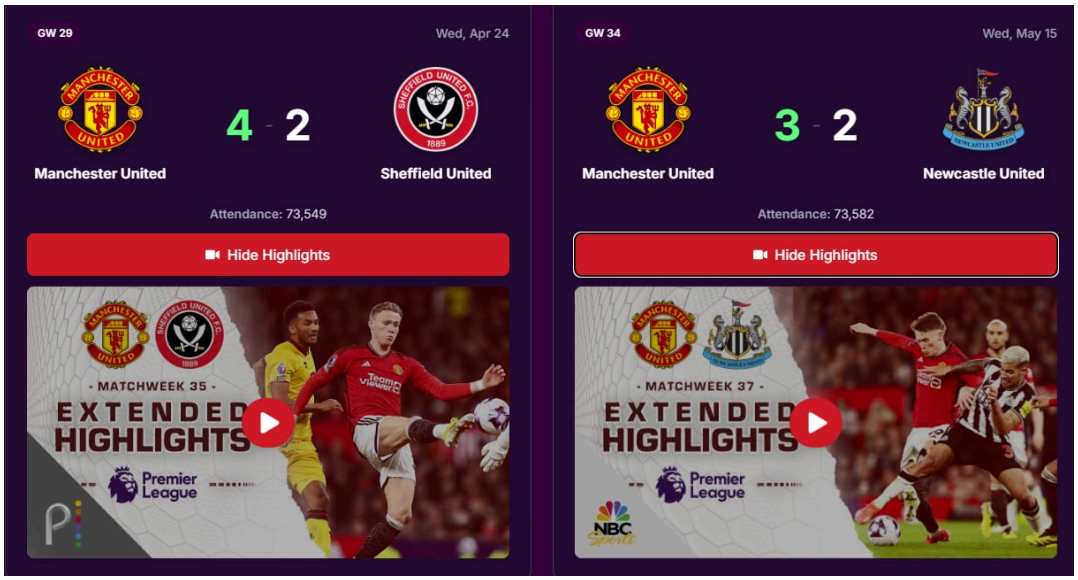
---

3. Features We're Actually Proud Of

3.1 The "Lite" YouTube Highlight System

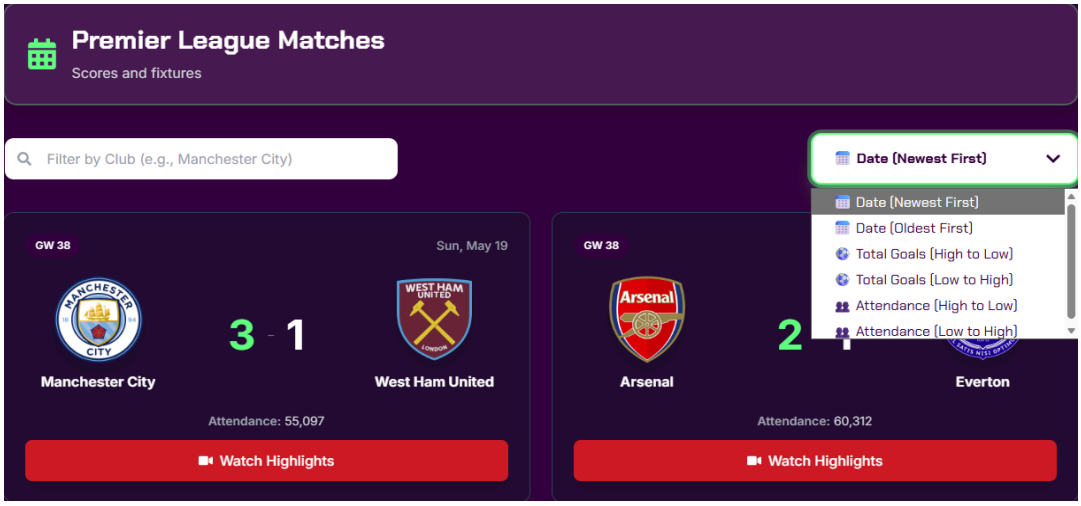
In our Matches.jsx page, we want people to be able to watch a match's highlights. However, storing 380 long URLs is messy and inefficient for storage.

The Feature: We designed the database to store only the 11-character YouTube ID (like dQw4w9WgXcQ). Then, we wrote a function in our Python ETL that automatically strips everything from a full URL except that ID. On the frontend, we built a "Lite-Embed" component that only loads the video when the user clicks on it, keeping the site fast even with hundreds of matches.



3.2 Finding a Needle in a Haystack: Match Filters

With 380 matches in a season, scrolling is a terrible option to find the game you want to watch. In Matches.jsx, we implemented a custom sorting and filter hub. You can search for a specific club (e.g., Arsenal, Chelsea) or use the "Sort By" dropdown to show just the games you love. We also used a useMemo hook in React to ensure this filtering happened instantly without lagging the browser.



3.3 The "PSR" Point Deduction Logic

Unexpectedly, in this season, there were a few exceptions regarding final season standings. Apparently, Everton and Nottingham Forest both had their points deducted. Thus, we had to build a point_adjustments table so that the "Official Standings" would accurately reflect reality and the media.

15	Everton (-8)	38	13	9	16	40	51	-11	40	W W D W L
16	Brentford	38	10	9	19	56	65	-9	39	W L D W L
17	Nottingham Forest (-4)	38	9	9	20	49	67	-18	32	L L W L W
18	Luton Town	38	6	8	24	52	85	-33	26	L L D L L
19	Burnley	38	5	9	24	41	78	-37	24	W D L L L
20	Sheffield United	38	3	7	28	35	104	-69	16	L L L L L

** Points deducted by the Premier League for Profit and Sustainability Rule (PSR) breaches.*

---

4. Difficulties

The Story of the React Transition: Early on, **Nam Nguyen** thought about using Tableau as the main visualization tool. However, after realizing the Tableau free version does not support direct database connection, he switched to Power BI. Then, **Nam** spent days trying to build an interactive dashboard in Power BI, assuming that the final report could be embedded into the website. The dashboard tracked a handful of features, such as League Positions over the Season, Cumulative Goals Difference Tracker, and Win/Draw/Lose Distribution across different Venues. When the dashboard was finished, however, we ran into the second problem. That was, we realized that to embed those dashboards into our website, we needed a paid license version of Power BI, which is not granted to an ordinary student (.edu) Microsoft account.

In response to this setback, we spent a whole night sitting together, trying to translate DAX formulas (e.g., RANKX, CALCULATE) - a Power BI code that controls interactivity - into the Front End of the website in another way using the React-Chartjs-2 library. The almost impossible task at the moment turned out to be not as hard when **Harrison** thought of asking Gemini for idea assistance and Claude for code implementation. As it turned out, it only took us more than 2 hours to successfully migrate from Power BI to JavaScript. Although a few features were left out as a result of the library's limitations, the majority of the features of **Nam**'s Power BI prototype were preserved.

---

5. Human Effort vs. AI Assistance:

To be completely honest about AI usage, we treated this project like a construction site.

The Human Work (The Engineers/Architects): We were the Lead Engineers. AI can't "think" through a project. However, AI is extremely efficient and great at providing solutions that require extra hours of thinking and implementation. Thus, our collaboration turns out to be:

- AI didn't know the data was dirty. So, we had to find the bugs and asked it to provide different ways to clean them.
- We designed the ER diagram, decided how the entities should be related, and chose in what ways to implement the ER model. In implementing the ER model, we consulted AI for best practices and ways to deal with ambiguity. For example, in implementing the Point_Adjustment table, we did not know that an attribute type could be a table in RDBMS.
- We heavily used AI in coming up with DAX formulas when designing the Power BI prototype, since this is a realm that neither of us had experience with. However, we knew what to do, and we asked AI how to do it correctly and efficiently.
- In migrating from Power BI to JavaScript, we also took the complex Power BI logic and relied heavily on AI to recommend to us how we should code up those Power-BI features.
- AI was also in charge of tedious tasks, such as writing Tailwind CSS classes to make the dashboard look "EPL Purple," or generating the initial "boilerplate" for our React components.

In total, our combined effort was **20+ hours**.

---

6. Ethical, Social, and Legal Issues

If we released this to the public, we'd have two big concerns:

Copyright: We are using official Premier League club logos and names. While okay for a school project, we'd need to use generic names/icons if we ever monetized it.

Gambling Ethics: Our charts can be beneficial for gamblers, since they are good at showing which teams are over-performing. In the wrong hands, this kind of data is used to fuel gambling addictions. We'd need to add a disclaimer that this is for "Educational Fans/Enthusiasts" only.

---

7. Conclusion and Future Work

If given more time, we would definitely expand the platform to include **Player Comparison Tools** and a **Match Outcome Predictor** based on the historical "Form" data we've already collected in our SQL views.

The project successfully demonstrates how a well-designed relational database can serve as the foundation for complex, interactive web experiences. By moving logic from the frontend to the database layer, the application remains high-performing while providing insights that go far beyond a standard scoreboard.