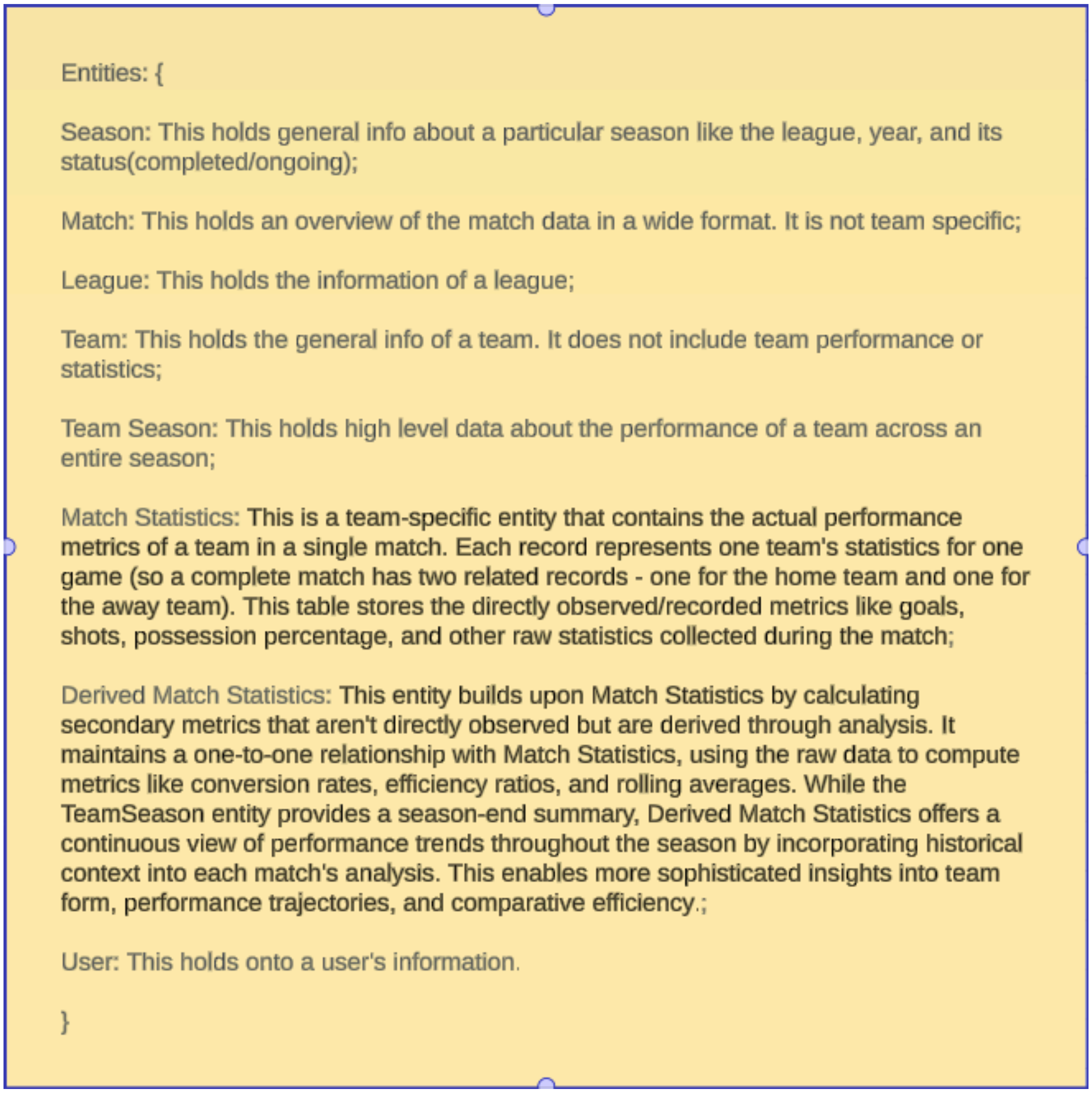


Entities: {

Season: This holds general info about a particular season like the league, year, and its status(completed/ongoing);

Match: This holds an overview of the match data in a wide format. It is not team specific;

League: This holds the information of a league;

Team: This holds the general info of a team. It does not include team performance or statistics;

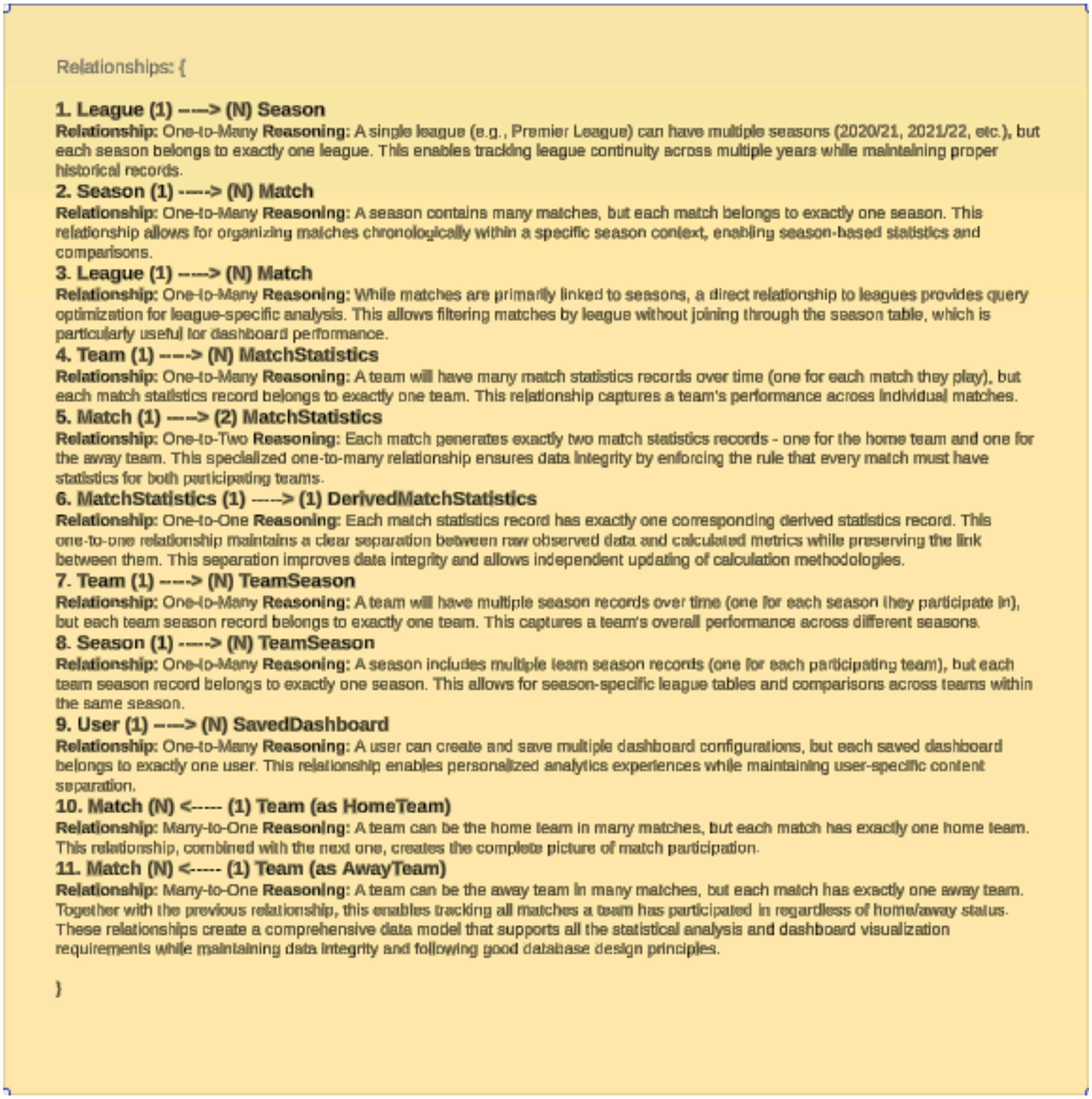
Team Season: This holds high level data about the performance of a team across an entire season;

Match Statistics: This is a team-specific entity that contains the actual performance metrics of a team in a single match. Each record represents one team's statistics for one game (so a complete match has two related records - one for the home team and one for the away team). This table stores the directly observed/recorded metrics like goals, shots, possession percentage, and other raw statistics collected during the match;

Derived Match Statistics: This entity builds upon Match Statistics by calculating secondary metrics that aren't directly observed but are derived through analysis. It maintains a one-to-one relationship with Match Statistics, using the raw data to compute metrics like conversion rates, efficiency ratios, and rolling averages. While the TeamSeason entity provides a season-end summary, Derived Match Statistics offers a continuous view of performance trends throughout the season by incorporating historical context into each match's analysis. This enables more sophisticated insights into team form, performance trajectories, and comparative efficiency.;

User: This holds onto a user's information.

}



Relationships: {

1. League (1) ----> (N) Season

Relationship: One-to-Many **Reasoning:** A single league (e.g., Premier League) can have multiple seasons (2020/21, 2021/22, etc.), but each season belongs to exactly one league. This enables tracking league continuity across multiple years while maintaining proper historical records.

2. Season (1) -----> (N) Match

Relationship: One-to-Many **Reasoning:** A season contains many matches, but each match belongs to exactly one season. This relationship allows for organizing matches chronologically within a specific season context, enabling season-based statistics and comparisons.

3. League (1) ----> (N) Match

Relationship: One-to-Many **Reasoning:** While matches are primarily linked to seasons, a direct relationship to leagues provides query optimization for league-specific analysis. This allows filtering matches by league without joining through the season table, which is particularly useful for dashboard performance.

4. Team (1) ----> (N) MatchStatistics

Relationship: One-to-Many **Reasoning:** A team will have many match statistics records over time (one for each match they play), but each match statistics record belongs to exactly one team. This relationship captures a team's performance across individual matches.

5. Match (1) ----> (2) MatchStatistics

Relationship: One-to-Two **Reasoning:** Each match generates exactly two match statistics records - one for the home team and one for the away team. This specialized one-to-many relationship ensures data integrity by enforcing the rule that every match must have statistics for both participating teams.

6. MatchStatistics (1) -----> (1) DerivedMatchStatistics

Relationship: One-to-One **Reasoning:** Each match statistics record has exactly one corresponding derived statistics record. This one-to-one relationship maintains a clear separation between raw observed data and calculated metrics while preserving the link between them. This separation improves data integrity and allows independent updating of calculation methodologies.

7. Team (1) ----> (N) TeamSeason

Relationship: One-to-Many **Reasoning:** A team will have multiple season records over time (one for each season they participate in), but each team season record belongs to exactly one team. This captures a team's overall performance across different seasons.

8. Season (1) -----> (N) TeamSeason

Relationship: One-to-Many **Reasoning:** A season includes multiple team season records (one for each participating team), but each team season record belongs to exactly one season. This allows for season-specific league tables and comparisons across teams within the same season.

9. User (1) ----> (N) SavedDashboard

Relationship: One-to-Many **Reasoning:** A user can create and save multiple dashboard configurations, but each saved dashboard belongs to exactly one user. This relationship enables personalized analytics experiences while maintaining user-specific content separation.

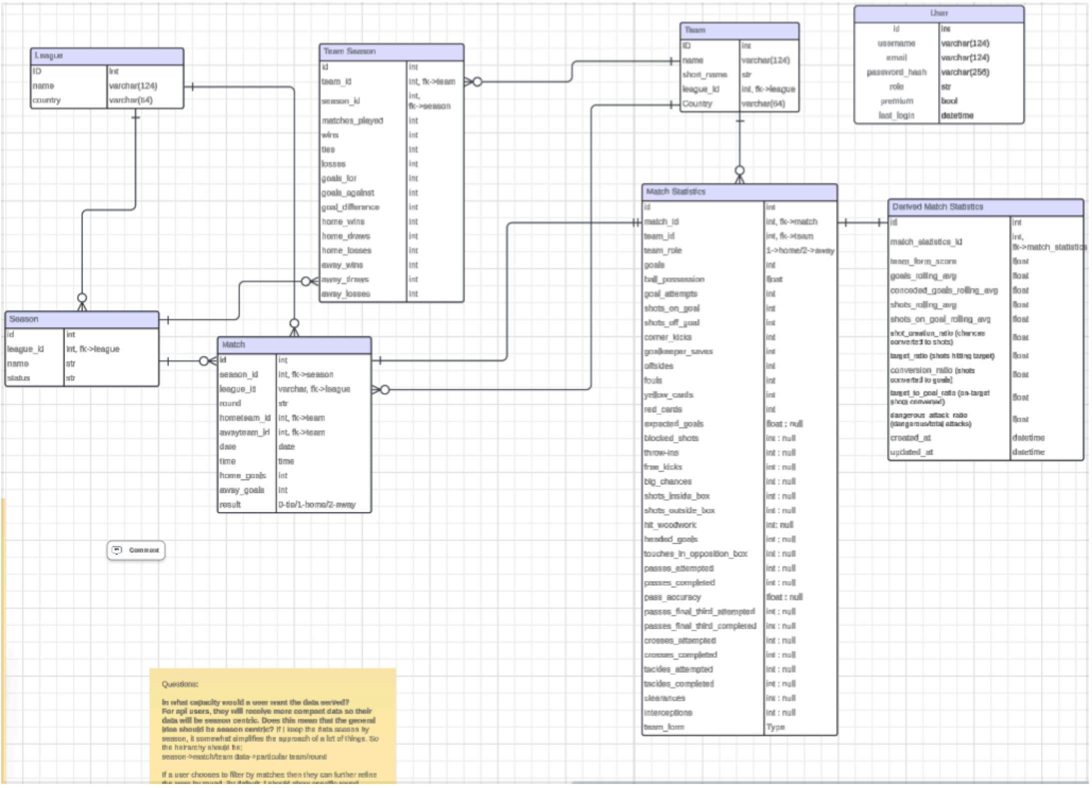
10. Match (N) <----- (1) Team (as HomeTeam)

Relationship: Many-to-One **Reasoning:** A team can be the home team in many matches, but each match has exactly one home team. This relationship, combined with the next one, creates the complete picture of match participation.

11. Match (N) <----- (1) Team (as AwayTeam)

Relationship: Many-to-One **Reasoning:** A team can be the away team in many matches, but each match has exactly one away team. Together with the previous relationship, this enables tracking all matches a team has participated in regardless of home/away status. These relationships create a comprehensive data model that supports all the statistical analysis and dashboard visualization requirements while maintaining data integrity and following good database design principles.

}



League	
ID	int
name	varchar(124)
country	varchar(64)

Team Season	
id	int
team_id	int, fk->team
season_id	int, fk->season
matches_played	int
wins	int
ties	int
losses	int
goals_for	int
goals_against	int
goal_difference	int
home_wins	int
home_draws	int
home_losses	int
away_wins	int
away_draws	int
away_losses	int

Team	
ID	int
name	varchar(124)
short_name	str
league_id	int, fk->league
Country	varchar(64)

User	
id	int
username	varchar(124)
email	varchar(124)
password_hash	varchar(256)
role	str
premium	bool
last_login	datetime

Season	
id	int
league_id	int, fk->league
name	str
status	str

Match	
id	int
season_id	int, fk->season
league_id	varchar, fk->league
round	str
hometeam_id	int, fk->team
awayteam_id	int, fk->team
date	date
time	time
home_goals	int
away_goals	int
result	0-tie/1-home/2-away

Match Statistics	
id	int
match_id	int, fk->match
team_id	int, fk->team
team_role	1->home/2->away
goals	int
ball_possession	float
goal_attempts	int
shots_on_goal	int
shots_off_goal	int
corner_kicks	int
goalkeeper_saves	int
offsides	int
fouls	int
yellow_cards	int
red_cards	int
expected_goals	float : null
blocked_shots	int : null
throw-ins	int : null
free_kicks	int : null
big_chances	int : null
shots_inside_box	int : null
shots_outside_box	int : null
hit_woodwork	int: null
headed_goals	int : null
touches_in_opposition_box	int : null
passes_attempted	int : null
passes_completed	int : null
pass_accuracy	float : null
passes_final_third_attempted	int : null
passes_final_third_completed	int : null
crosses_attempted	int : null
crosses_completed	int : null
tackles_attempted	int : null
tackles_completed	int : null
clearances	int : null
interceptions	int : null
team_form	Type

Derived Match Statistics	
id	int
match_statistics_id	int, fk->match_statistics
team_form_score	float
goals_rolling_avg	float
conceded_goals_rolling_avg	float
shots_rolling_avg	float
shots_on_goal_rolling_avg	float
shot_creation_ratio (chances converted to shots)	float
target_ratio (shots hitting target)	float
conversion_ratio (shots converted to goals)	float
target_to_goal_ratio (on-target shots converted)	float
dangerous_attack_ratio (dangerous/total attacks)	float
created_at	datetime
updated_at	datetime

Questions:

In what capacity would a user want the data served?
For api users, they will receive more compact data so their data will be season centric. Does this mean that the general idea should be season centric? If I keep the data season by season, it somewhat simplifies the approach of a lot of things. So the hairarchy should be:
season->match/team data->particular team/round

If a user chooses to filter by matches then they can further refine the [illegible] by round. By default, I should show specific round