# Project Phase 3 – Relational Database of Soccer/Football ER Diagram

**Team: Stormers**

By:

Nirmit Dagli

Srujan Sai Vemula

Nicholas Darko Brown

## 1. DESCRIPTION OF USE CASES

### 1. Comprehensive Player Performance Analysis Use Case: Technical Analysis for Player Evaluation

- Primary Users: Technical Directors, Scouts, Managers
- Purpose: Evaluate overall player performance across multiple metrics
- Business Value:
    - Identify top performers for potential recruitment
    - Assess current squad performance
    - Make informed decisions about contract renewals
    - Support team selection decisions
- Key Metrics:
    - Overall contribution (goals + assists)
    - Match participation
    - Passing ability
    - Physical performance (distance covered)
- Filtering Logic:
    - Minimum 3 matches played (ensures statistical significance)
    - Minimum 3 goal contributions (focuses on impactful players)

2. **Young Talent Analysis Use Case: Youth Talent Scouting and Development**

- Primary Users: Scouts, Youth Development Teams, Recruitment Teams
- Purpose: Identify and track promising young players
- Business Value:
    - Early identification of talent
    - Investment opportunities in young players
    - Long-term squad planning
    - Academy performance assessment
- Key Metrics:
    - Age verification
    - Playing time analysis
    - Production metrics (goals/assists)
    - Technical ability (passing stats)
- Filtering Logic:
    - Born after 2001 (under 23)
    - Minimum 2 matches and 60 minutes per game (regular players)

3. **Team Performance Analysis Use Case: Team Performance Monitoring and Reporting**

- Primary Users: Club Management, Coaches, Media Teams
- Purpose: Analyze team's overall performance and trends
- Business Value:
    - Track team progress
    - Compare performance against targets
    - Stadium utilization analysis
    - Financial planning (attendance trends)
- Key Metrics:
    - Win/Draw/Loss record
    - Goal difference
    - Average attendance
    - Points calculation (3 for win, 1 for draw)
- Filtering Logic:
    - Minimum 3 matches played (ensures meaningful analysis)
    - Complete game records (wins + draws + losses >= 3)

4. **Injury Impact Analysis Use Case: Medical and Performance Impact Assessment**

- Primary Users: Medical Staff, Fitness Coaches, Team Managers
- Purpose: Analyze how injuries affect player performance
- Business Value:
    - Improve injury prevention strategies
    - Optimize return-to-play protocols
    - Monitor player recovery effectiveness
    - Risk assessment for player availability
- Key Metrics:
    - Injury frequency
    - Recovery time analysis
    - Post-injury performance
    - Playing time management
- Filtering Logic:
    - Has recorded injuries
    - Minimum 2 matches after return (performance assessment)

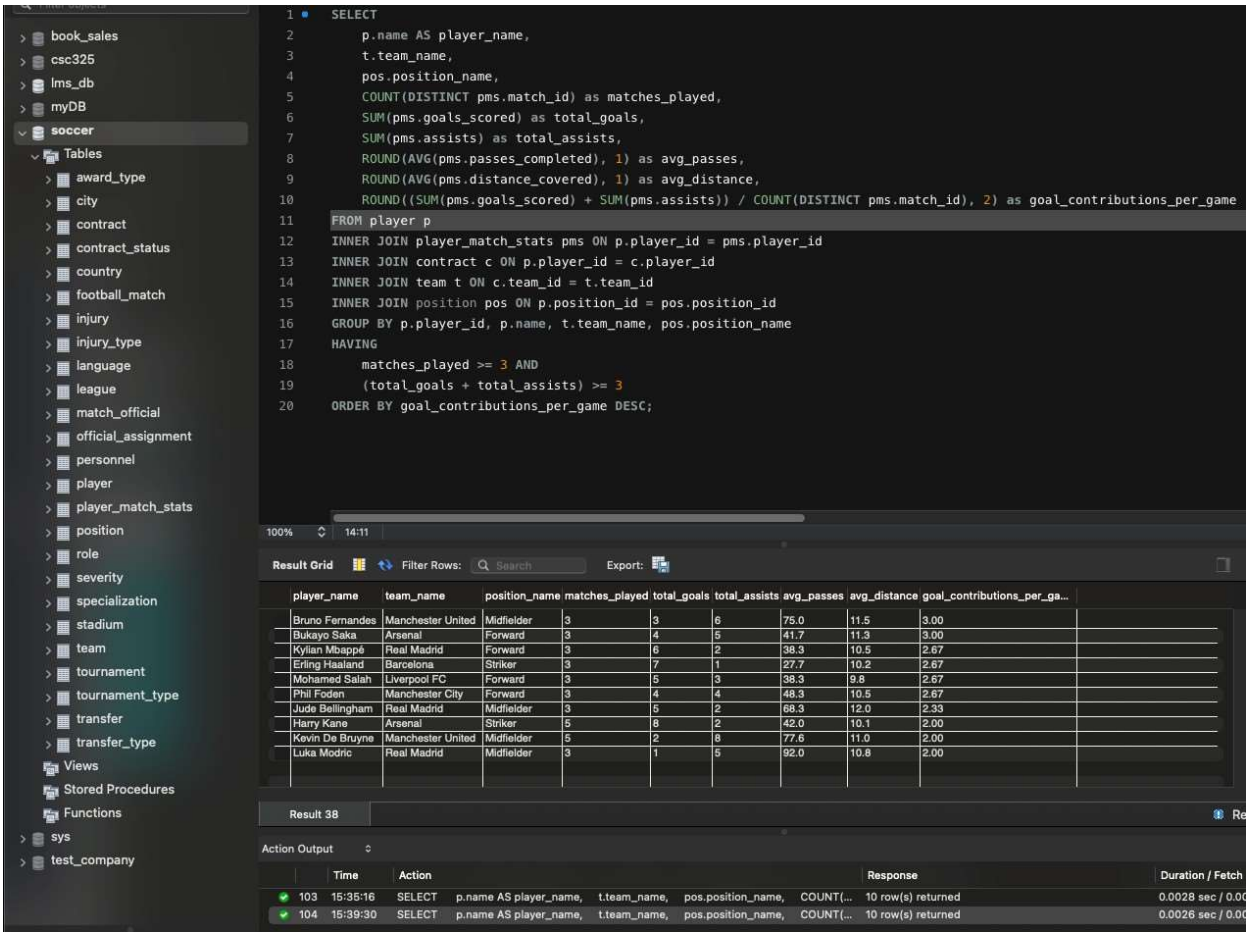
5. **Playmaker Performance Analysis Use Case: Creative Player Assessment**

- Primary Users: Tactical Analysts, Coaches, Recruitment Team
- Purpose: Evaluate players' creative and assistive contributions
- Business Value:
    - Tactical planning
    - Identify key playmakers
    - Recruitment planning
    - Team style analysis
- Key Metrics:
    - Assist numbers
    - Passing efficiency
    - Creative contribution
    - Work rate (distance covered)
- Filtering Logic:
    - Only midfielders and forwards
    - Minimum 3 matches played
    - Minimum 2 assists
    - Ordered by creative contribution

## 2. SQL QUERIES FOR USE CASES

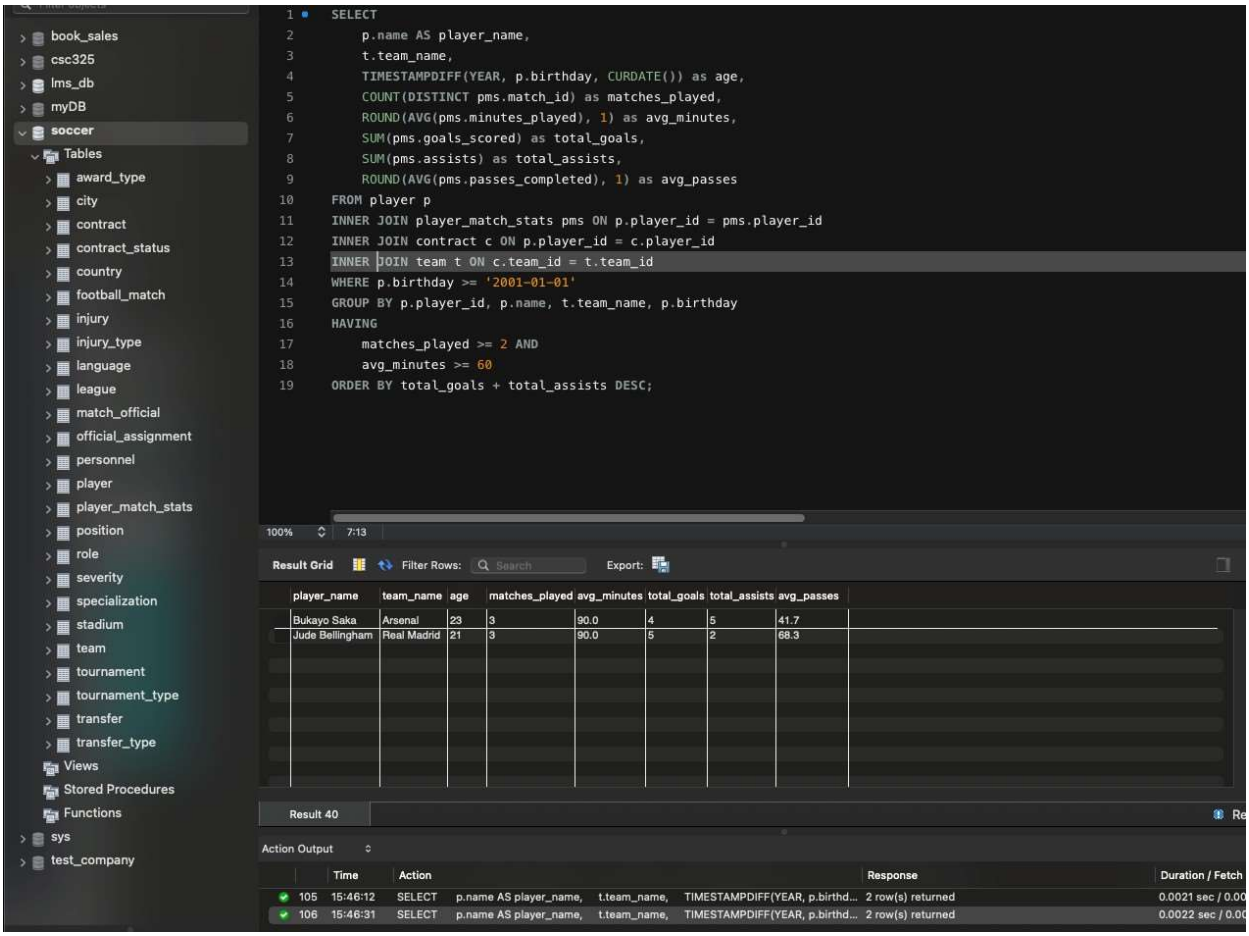
1. SELECT
   p.name AS player_name,
   t.team_name,
   pos.position_name,
   COUNT(DISTINCT pms.match_id) as matches_played,
   SUM(pms.goals_scored) as total_goals,
   SUM(pms.assists) as total_assists,
   ROUND(AVG(pms.passes_completed), 1) as avg_passes,
   ROUND(AVG(pms.distance_covered), 1) as avg_distance,
   ROUND((SUM(pms.goals_scored) + SUM(pms.assists)) / COUNT(DISTINCT pms.match_id), 2) as goal_contributions_per_game
FROM player p
INNER JOIN player_match_stats pms ON p.player_id = pms.player_id
INNER JOIN contract c ON p.player_id = c.player_id
INNER JOIN team t ON c.team_id = t.team_id
INNER JOIN position pos ON p.position_id = pos.position_id
GROUP BY p.player_id, p.name, t.team_name, pos.position_name
HAVING
   matches_played >= 3 AND
   (total_goals + total_assists) >= 3
ORDER BY goal_contributions_per_game DESC;

# Player Performance Analysis



2. SELECT

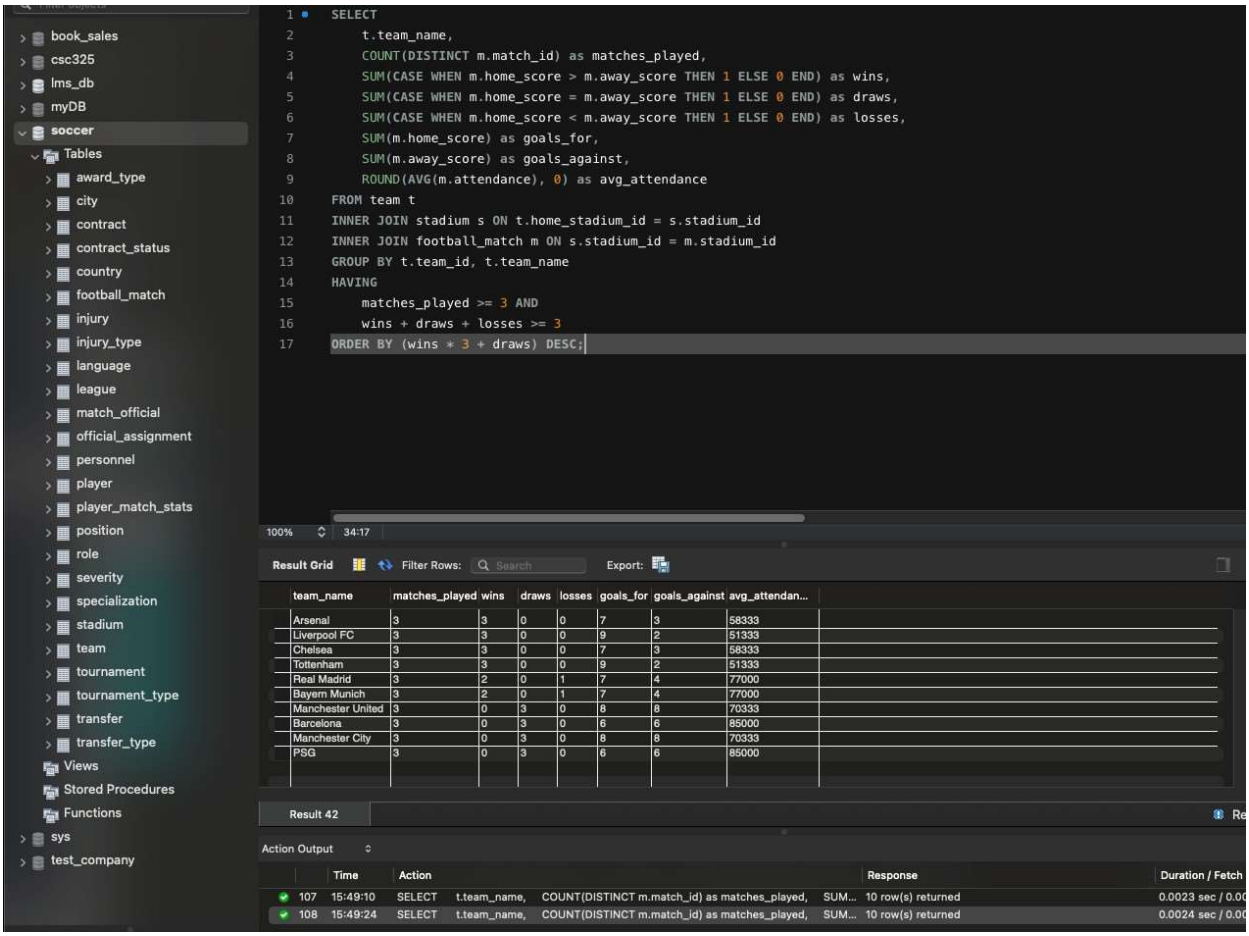
```
    p.name AS player_name,
    t.team_name,
    TIMESTAMPDIFF(YEAR, p.birthday, CURDATE()) as age,
    COUNT(DISTINCT pms.match_id) as matches_played,
    ROUND(AVG(pms.minutes_played), 1) as avg_minutes,
    SUM(pms.goals_scored) as total_goals,
    SUM(pms.assists) as total_assists,
    ROUND(AVG(pms.passes_completed), 1) as avg_passes
FROM player p
INNER JOIN player_match_stats pms ON p.player_id = pms.player_id
INNER JOIN contract c ON p.player_id = c.player_id
INNER JOIN team t ON c.team_id = t.team_id
WHERE p.birthday >= '2001-01-01'
GROUP BY p.player_id, p.name, t.team_name, p.birthday
HAVING
    matches_played >= 2 AND
    avg_minutes >= 60
ORDER BY total_goals + total_assists DESC;
```




Young Talent Analysis (Players under 23)

3. SELECT

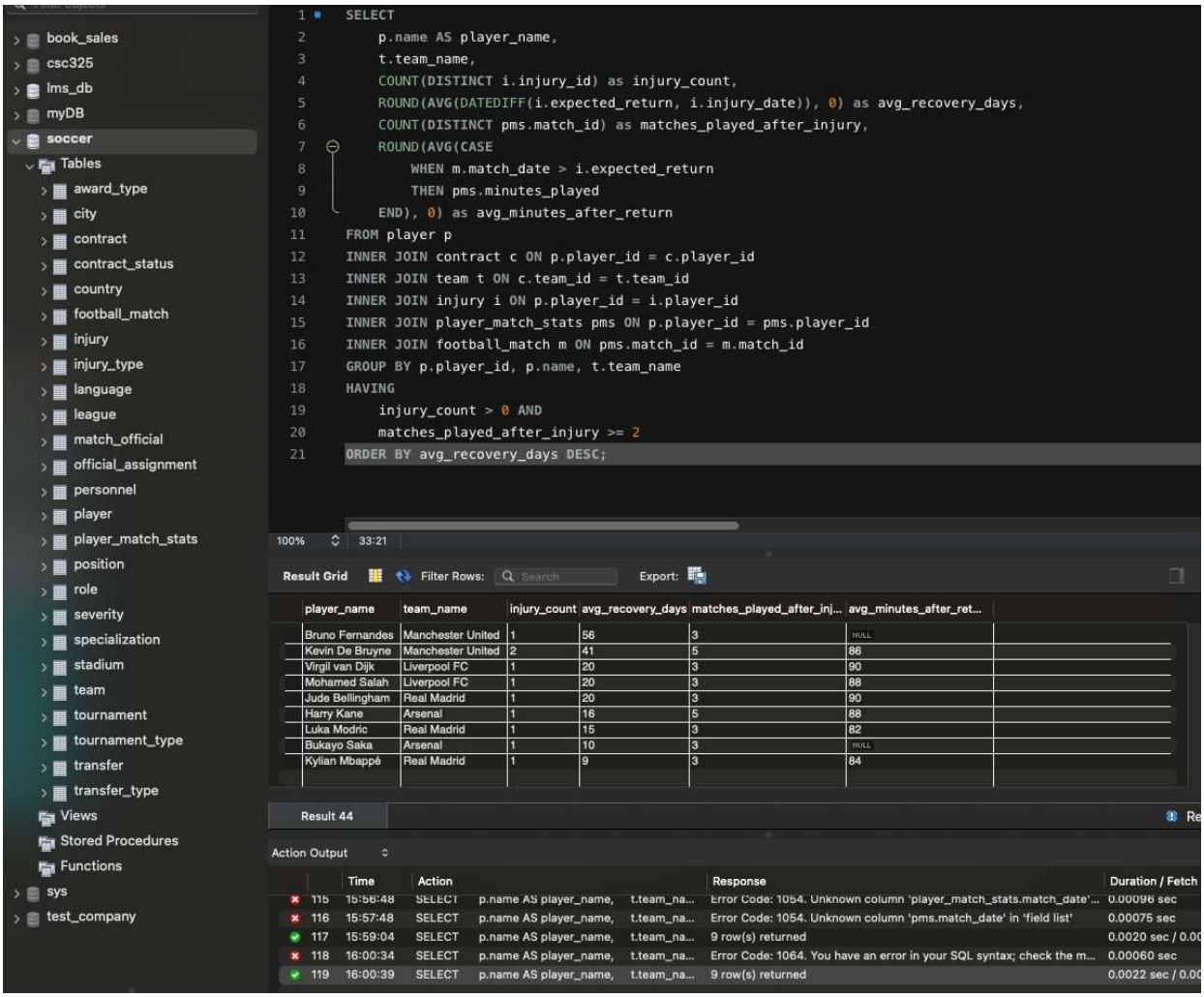
```
    t.team_name,
    COUNT(DISTINCT m.match_id) as matches_played,
    SUM(CASE WHEN m.home_score > m.away_score THEN 1 ELSE 0 END) as wins,
    SUM(CASE WHEN m.home_score = m.away_score THEN 1 ELSE 0 END) as draws,
    SUM(CASE WHEN m.home_score < m.away_score THEN 1 ELSE 0 END) as losses,
    SUM(m.home_score) as goals_for,
    SUM(m.away_score) as goals_against,
    ROUND(AVG(m.attendance), 0) as avg_attendance
FROM team t
INNER JOIN stadium s ON t.home_stadium_id = s.stadium_id
INNER JOIN football_match m ON s.stadium_id = m.stadium_id
GROUP BY t.team_id, t.team_name
HAVING
    matches_played >= 3 AND
    wins + draws + losses >= 3
ORDER BY (wins * 3 + draws) DESC;
```




Team Performance Analysis

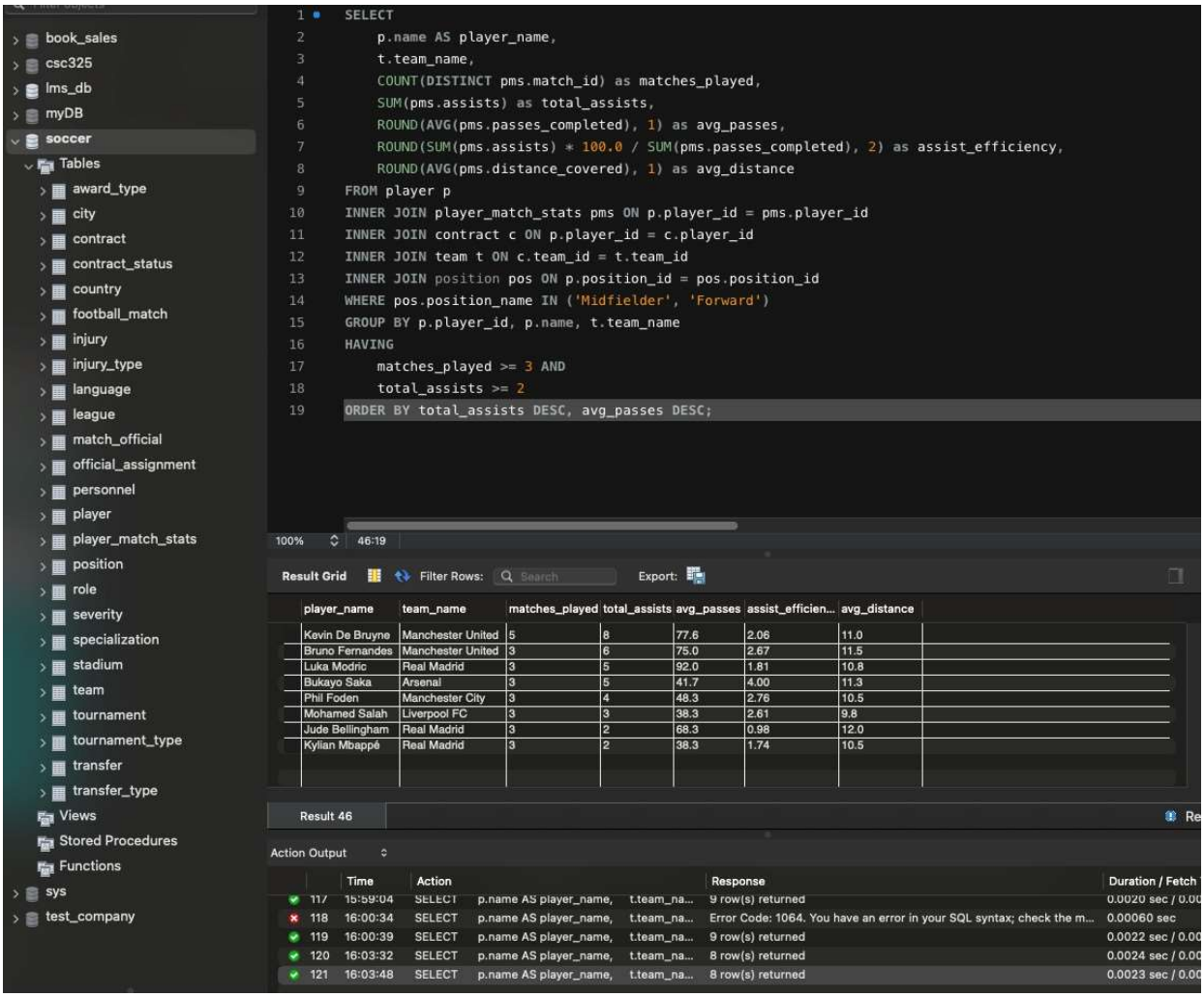
4. SELECT

```
    p.name AS player_name,
    t.team_name,
    COUNT(DISTINCT i.injury_id) as injury_count,
    ROUND(AVG(DATEDIFF(i.expected_return, i.injury_date)), 0) as avg_recovery_days,
    COUNT(DISTINCT pms.match_id) as matches_played_after_injury,
    ROUND(AVG(CASE
        WHEN m.match_date > i.expected_return
        THEN pms.minutes_played
    END), 0) as avg_minutes_after_return
FROM player p
INNER JOIN contract c ON p.player_id = c.player_id
INNER JOIN team t ON c.team_id = t.team_id
INNER JOIN injury i ON p.player_id = i.player_id
INNER JOIN player_match_stats pms ON p.player_id = pms.player_id
INNER JOIN football_match m ON pms.match_id = m.match_id
GROUP BY p.player_id, p.name, t.team_name
HAVING
    injury_count > 0 AND
    matches_played_after_injury >= 2
ORDER BY avg_recovery_days DESC;
```




Injury Impact on Performance

```
5. SELECT
   p.name AS player_name,
   t.team_name,
   COUNT(DISTINCT pms.match_id) as matches_played,
   SUM(pms.assists) as total_assists,
   ROUND(AVG(pms.passes_completed), 1) as avg_passes,
   ROUND(SUM(pms.assists) * 100.0 / SUM(pms.passes_completed), 2) as assist_efficiency,
   ROUND(AVG(pms.distance_covered), 1) as avg_distance
FROM player p
INNER JOIN player_match_stats pms ON p.player_id = pms.player_id
INNER JOIN contract c ON p.player_id = c.player_id
INNER JOIN team t ON c.team_id = t.team_id
INNER JOIN position pos ON p.position_id = pos.position_id
WHERE pos.position_name IN ('Midfielder', 'Forward')
GROUP BY p.player_id, p.name, t.team_name
HAVING
   matches_played >= 3 AND
   total_assists >= 2
ORDER BY total_assists DESC, avg_passes DESC;
```




Playmaker Performance Analysis

## MongoDB Design Reasoning

The updated MongoDB design uses a unified document structure that brings all related information about a player into a single document. This means that details like personal information, team affiliation, match performance, contracts, injuries, and transfer history are all stored together in one place. For example, a player's match statistics—such as goals scored, assists, and minutes played—are embedded directly within their document, along with team details like the team's name, jersey color, and market value.

By integrating everything into one document, we can retrieve all the data about a player with just one query. This is especially useful for scenarios like scouting reports, performance evaluations, or team management, where all this information is often needed at once. Instead of making multiple queries or joining collections, we get everything in one go, which saves time and improves performance.

This design works well for our use case because MongoDB's schema-less nature allows us to easily adapt and add new fields as needed. For instance, if we wanted to track advanced analytics or player-specific metrics in the future, we could do so without having to redesign the database. While storing all this data in one document might create some redundancy—such as repeating team details for multiple players—it keeps the queries simpler and faster, which is more important for the kind of read-heavy operations we anticipate.

In summary, the unified document approach makes the database efficient, flexible, and easy to work with, especially for real-time access to interrelated data. It's a practical choice for managing the complex and hierarchical nature of sports data.

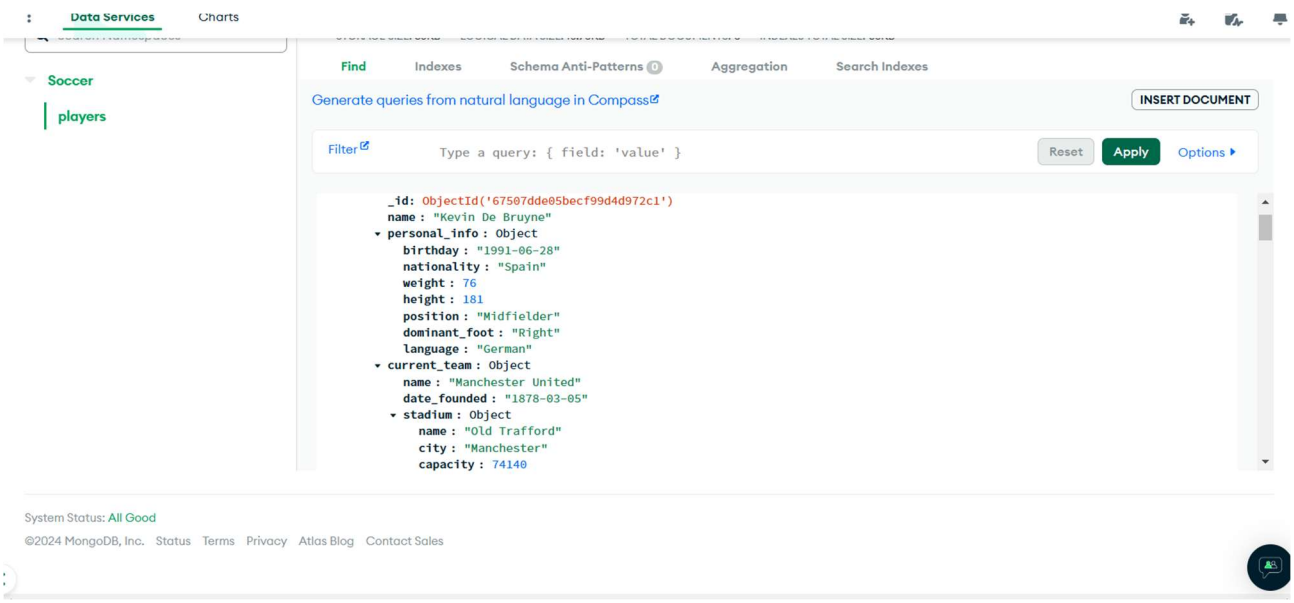
### MongoDB Schema Design

- **Key Entity:** Player
    - Reason: Players are the central unit of analysis and the most dynamic entity, with attributes that frequently change, such as match statistics, injuries, and transfers.
- **Collections:**
    - **Player:** Contains player details such as personal info, current team, contract, match stats, injuries, and tournaments.
    - **Team (embedded in Player):** Stores team-related details like name, stadium, league, and market value.
    - **Match Stats (array in Player):** Includes per-match data like goals scored, minutes played, assists, and results.

- **Tournaments (array in Player):** Captures tournament details such as type (league, cup), start/end dates, and prizes.
- **Injuries (array in Player):** Tracks injury severity, recovery plan, and injury date.
- **Transfers (array in Player):** Records transfer details like type (permanent/loan), teams involved, and transfer fees.

**Design Choices and Reasoning**

- **Player-Centric Design:**
  - Players are the most dynamic entities, and their data is accessed more frequently than teams.
  - A player-centric approach avoids fragmentation caused by frequent transfers and injuries.
- **Embedded Documents:**
  - Match stats, injuries, and contracts are embedded within the player document for faster read operations.
- **Use of Arrays:**
  - Arrays are used to store repeating attributes like match stats, injuries, transfers, and tournaments, ensuring data is logically grouped.
- **Normalization and Referencing:**
  - Key data like current team details are embedded for quick access.
  - Static team attributes like stadiums and leagues are referenced where possible to avoid redundancy.




MongoDB Sample Collection

**Each Team Member's Contribution**

1. **Nirmit Dagli:**
    - MongoDB Schema Design: Created a player-centric database design with embedded documents (implementation of MongoDB Atlas).
    - Implemented match stats and injury tracking.
    - Coordinated team tasks and ensured timely completion.
    - Generated outputs and captured screenshots.

2. **Nicholas Brown:**
    - SQL Queries: Designed and executed SQL queries for retrieving player stats, injuries, and team details.
    - Generated outputs and captured screenshots for use case demonstrations.
    - Ensured query optimization for large datasets.

3. **Sai Srujan Vemula:**
    - Project and PPT Documentation: Compiled the report
    - Tested queries for performance and accuracy
    - MongoDB and SQL Schema Design
    - Analyzed obstacles and documented the learning experience.