

Sports Tournament Registration Platform

Course: 4707.501

Instructor: Tozammel Hossain

Date: 12/09/2025

Group 3: Kevin Ho, Ngan ‘Kacie’ Do & Gohar Kirakosyan

TABLE OF CONTENTS

1. Introduction-----	3
2. Database Design-----	3
2.1. Objectives and Scope-----	3
2.2. User Requirements-----	3
2.3. Business Rules-----	4
2.4. ERD Description-----	4
2.5. Data Dictionary-----	5
3. Database Implementation-----	7
3.1. Database Environment-----	7
3.2. Database Schema Creation-----	7
3.3. Data Insertion-----	9
3.4. Data Retrieval and Analytical Reports-----	13
4. Conclusion-----	17

1. Introduction

The purpose of this project is to design, implement, and demonstrate a fully functional relational database for a Sports Tournament Registration Platform. This platform is designed to assist universities, sports clubs, and tournament organizers in managing sports, tournaments, matches, team registrations, player information, venues, and match results.

This report presents revised documentation from Phases 1 and 2, followed by implementation details, including SQL table creation, sample data population, and data retrieval queries.

2. Database Design

2.1. Objectives and Scope

The primary objective of the Sports Tournament Registration Platform is to provide a normalized and efficient data model capable of supporting registration processes, match scheduling, and analytical reporting.

Objectives:

- Design a normalized relational model to store sports, tournaments, teams, players, matches, and results.
- Provide queries and reports for organizers, coaches, and public viewers.
- Implement the final database in MySQL Workbench using structured SQL scripts.
- Populate tables with sample data for demonstration and testing.
- Demonstrate queries that return meaningful analytical insights.

Scope:

The system targets university-level sports departments, intramural sports leagues, and event organizers. Its functional boundaries include player registration, match creation, tournament management, and public information queries. The system does not include payment processing, live scoring interfaces, or mobile UI development.

2.2. User Requirements

The system supports three main user groups: Tournament Organizers, Teams/Players, and Spectators/Public.

Tournament Organizers:

- Create, edit, and delete tournaments.
- View tournaments

- Schedule and view matches
- Register teams for tournaments.
- Update match results.
- Generate reports such as department participation and sport participation reports.

Teams & Players:

- View schedules, tournaments, matches
- Register teams for upcoming tournaments.
- Add and remove players before roster freeze.
- View historical performance.

Spectators:

- View match results
- View upcoming matches

These requirements guided the database design and the implementation of SQL-based reporting.

2.3. Business Rules

1. A sport may have multiple tournaments, but each tournament belongs to exactly one sport.
2. A team may register for multiple tournaments, and a tournament may include multiple teams.
3. A player belongs to exactly one team but may participate in multiple tournaments through that team.
4. A match always involves exactly two teams and takes place at one venue.
5. A referee may officiate many matches, but a match has at most one referee.
6. Each match must belong to one tournament.
7. Scores may only be entered once a match is completed.
8. A team's win/loss record is automatically derived from match results.

2.4. ERD Description

The Entity-Relationship Diagram (ERD) includes the following main entities:

- Sport – Defines categories of sports.
- Tournament – Represents a competition under a sport.

- Team – Represents a competitive group participating in tournaments.
- Player – Stores roster information for each team.
- Match – Defines individual games between two teams.
- Venue – Indicates where matches occur.
- Referee – Officials assigned to matches.
- TeamTournament – Junction table linking teams to tournaments.

The ERD supports 1-to-many, many-to-many, and identifying relationships essential for tournament tracking.

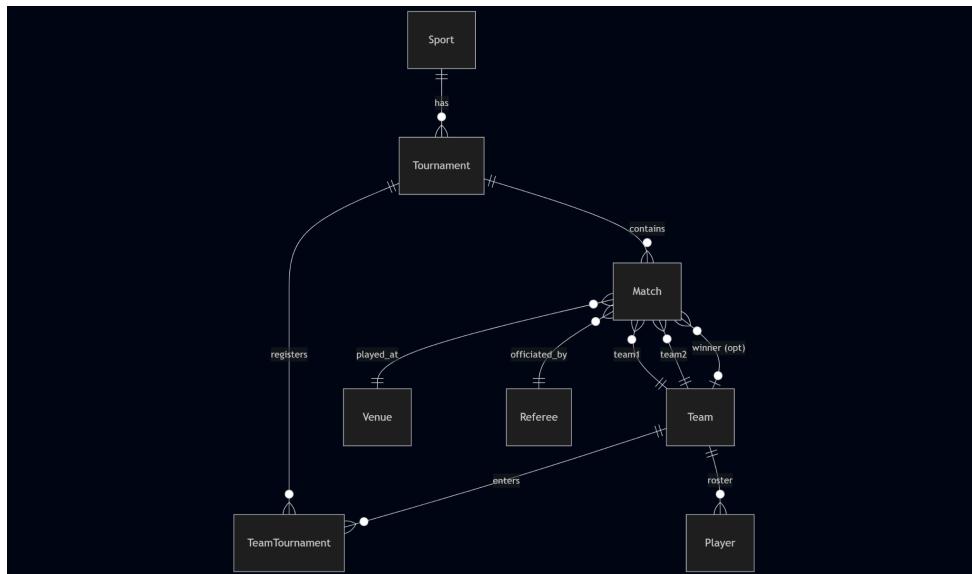


Figure 1: Entity-Relationship Diagram

2.5. Data Dictionary

Entity	Attribute	Data Type	Description
Sport	SportID (PK)	INT	Unique identifier for sport
	SportName	VARCHAR(50)	Name of the sport (e.g., Basketball, Volleyball)
Tournament	TournamentID (PK)	INT	Unique identifier for tournament
	SportID (FK)	INT	Associated sport
	Name	VARCHAR(100)	Tournament name

	StartDate	DATE	Tournament start date
	EndDate	DATE	Tournament end date
	MaxTeams	INT	Maximum number of teams allowed
Team	TeamID (PK)	INT	Unique team ID
	TeamName	VARCHAR(100)	Team name
	Department	VARCHAR(50)	Department affiliation
	Coach	VARCHAR(100)	Team coach name
Player	PlayerID (PK)	INT	Unique player ID
	TeamID (FK)	INT	Associated team
	PlayerName	VARCHAR(100)	Player's full name
	Department	VARCHAR(50)	Department or major
	YearLevel	VARCHAR(10)	Player's academic year
Venue	VenueID (PK)	INT	Unique venue ID
	VenueName	VARCHAR(100)	Name of venue
	Location	VARCHAR(100)	Venue address or campus location
Referee	RefereeID (PK)	INT	Unique referee ID
	RefereeName	VARCHAR(100)	Referee's name
	Qualification	VARCHAR(100)	Certification or level
Match	MatchID (PK)	INT	Unique match ID
	TournamentID (FK)	INT	Associated tournament
	Team1ID (FK)	INT	First participating team
	Team2ID (FK)	INT	Second participating

			team
	VenueID (FK)	INT	Venue for the match
	RefereeID (FK)	INT	Assigned referee
	MatchDate	DATETIME	Scheduled date and time
	Team1Score	INT	Score for team 1
	Team2Score	INT	Score for team 2
	WinnerTeamID (FK)	INT	Winning team
Teamtournamet	TeamID (PK)	INT	ID of team
	TournamentID (PK)	INT	ID of tournament

Each attribute was selected to properly support user requirements, minimize redundancy, and satisfy integrity constraints.

3. Database Implementation

3.1. Database Environment

DBMS used: MySQL WorkBench 8.0

SQL scripts are executed in Jupyter Notebook to ensure ease of verification.

3.2. Database Schema Creation

The database concludes 8 entities. Each main entity contains at least 25 records, while others contain at least 5 records.

Below are example SQL statements used to create tables:

```
[76]: from sqlalchemy import text
tables_sql = """
CREATE TABLE IF NOT EXISTS Sport (
    SportID SERIAL PRIMARY KEY,
    SportName VARCHAR(50) NOT NULL
);

CREATE TABLE IF NOT EXISTS Tournament (
    TournamentID SERIAL PRIMARY KEY,
    SportID INT REFERENCES Sport(SportID),
    Name VARCHAR(100) NOT NULL,
    StartDate DATE NOT NULL,
    EndDate DATE NOT NULL,
    MaxTeams INT NOT NULL
);

CREATE TABLE IF NOT EXISTS Team (
    TeamID SERIAL PRIMARY KEY,
    TeamName VARCHAR(100),
    Department VARCHAR(50),
    Coach VARCHAR(100)
);

CREATE TABLE IF NOT EXISTS Player (
    PlayerID SERIAL PRIMARY KEY,
    TeamID INT REFERENCES Team(TeamID),
    PlayerName VARCHAR(100),
    Department VARCHAR(50),
    YearLevel VARCHAR(10)
);
```

Figure 1: Creating Tables for Sport, Tournament, Team, & Players

```
CREATE TABLE IF NOT EXISTS Venue (
    VenueID SERIAL PRIMARY KEY,
    VenueName VARCHAR(100),
    Location VARCHAR(100)
);

CREATE TABLE IF NOT EXISTS Referee (
    RefereeID SERIAL PRIMARY KEY,
    RefereeName VARCHAR(100),
    Qualification VARCHAR(100)
);

CREATE TABLE IF NOT EXISTS 'Match' (
    MatchID SERIAL PRIMARY KEY,
    TournamentID INT REFERENCES Tournament(TournamentID),
    Team1ID INT REFERENCES Team(TeamID),
    Team2ID INT REFERENCES Team(TeamID),
    VenueID INT REFERENCES Venue(VenueID),
    RefereeID INT REFERENCES Referee(RefereeID),
    MatchDate TIMESTAMP,
    Team1Score INT,
    Team2Score INT,
    WinnerTeamID INT REFERENCES Team(TeamID)
);

# CREATE TABLE IF NOT EXISTS TeamTournament (
#     TournamentID INT REFERENCES Tournament(TournamentID),
#     TeamID INT REFERENCES Team(TeamID),
#     PRIMARY KEY (TournamentID, TeamID)
# );
"""

with engine.begin() as conn:
    for statement in tables_sql.strip().split(';'):
        stmt = statement.strip()
        if stmt:
            conn.execute(text(stmt))
print("All tables created!")

All tables created!
```

Figure 2: Creating Tables for Venue, Referee, Match, and TeamTournament.

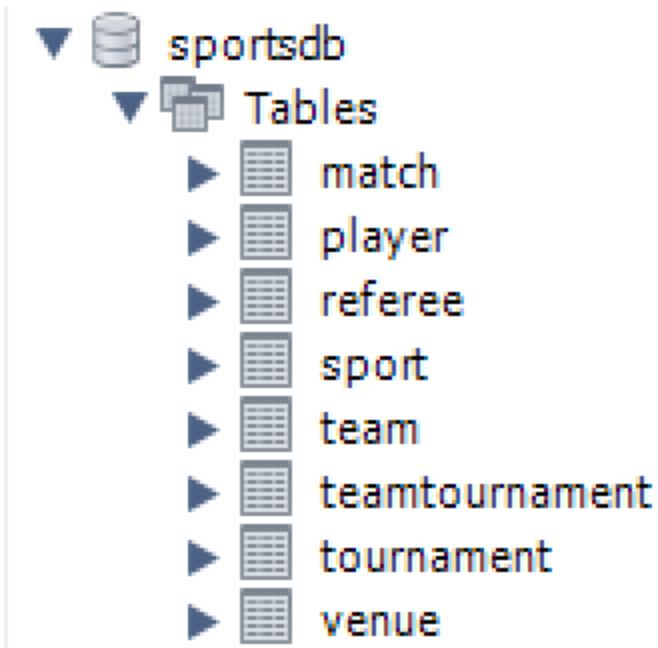


Figure 3: List of Tables created successfully.

3.3. Data Insertion

After tables were created, data was inserted using SQL INSERT INTO commands.

```
[78]: def import_data():

    print("Importing data...")

    # SPORT
    insert_sport ="""
        INSERT INTO Sport (SportName) VALUES
        ('Basketball'), ('Volleyball'), ('Soccer'),
        ('Badminton'), ('Table Tennis'), ('Tennis');
    """

    # VENUE
    insert_venue ="""
        INSERT INTO Venue (VenueName, Location) VALUES
        ('Main Gym A','Campus North'),
        ('Main Gym B','Campus North'),
        ('Outdoor Field 1','East Campus'),
        ('Outdoor Field 2','East Campus'),
        ('Court 1','Sports Complex'),
        ('Court 2','Sports Complex'),
        ('Poolside Arena','Aquatic Center'),
        ('Indoor Hall 1','Campus South'),
        ('Indoor Hall 2','Campus South'),
        ('Tennis Court 1','Sports Complex'),
        ('Tennis Court 2','Sports Complex'),
        ('Table Tennis Room','Recreation Center'),
        ('Badminton Court 1','Rec Center'),
        ('Badminton Court 2','Rec Center'),
        ('Soccer Pitch A','West Campus'),
        ('Soccer Pitch B','West Campus'),
        ('Volleyball Arena','Sports Complex'),
        ('Practice Field 1','Training Grounds'),
        ('Practice Field 2','Training Grounds'),
        ('Gym Annex','East Annex'),
        ('Multi-Purpose Hall','Central Campus'),
        ('Lecture Hall B','Central Campus'),
        ('Alumni Court','Alumni Center'),
        ('Referee Office','Admin Building'),
        ('Outdoor Arena','Parkside');
    """

    # REFEREE
    insert_referee ="""
        INSERT INTO Referee (RefereeName, Qualification) VALUES
        ('Alex Morgan','Level 2'),
        ('Brian Chan','Level 1'),
        ('Carla Reyes','Level 3'),
        ('Diego Alvarez','Level 1'),
        ('Emily Park','Level 2'),
        ('Farah Khan','Level 2'),
        ('George Li','Level 1'),
        ('Hannah Smith','Level 3');
    """

```

Figure 4: Insert data into SPORT, VENUE, REFEREE tables

```

# TEAM
insert_team = """
INSERT INTO Team (TeamName, Department, Coach) VALUES
('Falcons', 'Engineering', 'Coach A'),
('Tigers', 'Business', 'Coach B'),
('Lions', 'Arts', 'Coach C'),
('Wolves', 'Science', 'Coach D'),
('Panthers', 'Law', 'Coach E'),
('Sharks', 'Medicine', 'Coach F'),
('Rangers', 'Engineering', 'Coach G'),
('Knights', 'Business', 'Coach H'),
('Lions', 'Arts', 'Coach I'),
('Hawks', 'Science', 'Coach J'),
('Bulls', 'Law', 'Coach K'),
('Wizards', 'Medicine', 'Coach L'),
('Spartans', 'Engineering', 'Coach M'),
('Dragons', 'Business', 'Coach N'),
('Pirates', 'Arts', 'Coach O'),
('Vikings', 'Science', 'Coach P'),
('Titans', 'Law', 'Coach Q'),
('UFC', 'Engineering', 'Coach R'),
('Comets', 'Engineering', 'Coach S'),
('Stallions', 'Business', 'Coach T'),
('Warriors', 'Arts', 'Coach U'),
('Knights 2', 'Science', 'Coach V'),
('Coyotes', 'Law', 'Coach W'),
('Rockets', 'Medicine', 'Coach X'),
('Cyclones', 'Engineering', 'Coach Y');
"""

# PLAYER
insert_player = """
INSERT INTO Player (TeamID, PlayerName, Department, YearLevel) VALUES
(1, 'Alice Johnson', 'Engineering', '3'),
(2, 'Bob Smith', 'Business', '2'),
(3, 'Clara Gomez', 'Arts', '4'),
(4, 'David Kim', 'Science', '1'),
(5, 'Eva Long', 'Law', '3'),
(6, 'Frank Wu', 'Medicine', '2'),
(7, 'Grace Lee', 'Engineering', '4'),
(8, 'Harry Potter', 'Business', '1'),
(9, 'Ivy Green', 'Arts', '2'),
(10, 'Jack He', 'Science', '3'),
(11, 'Kara Z', 'Law', '4'),
(12, 'Leo W', 'Medicine', '2'),
(13, 'Maya P', 'Engineering', '1'),
(14, 'Nolan Q', 'Business', '3'),
(15, 'Olivia R', 'Arts', '2'),
(16, 'Parker S', 'Science', '4'),
(17, 'Quinn T', 'Law', '1'),
(18, 'Rita U', 'Medicine', '3'),
(19, 'Sam V', 'Engineering', '2'),
(20, 'Tina W', 'Business', '4'),
(21, 'Uma X', 'Arts', '1'),
(22, 'Victor Y', 'Science', '2'),
(23, 'Wendy Z', 'Law', '3'),
(24, 'Xavier A', 'Medicine', '1'),
(25, 'Yara B', 'Engineering', '3');
"""

```

Figure 5: Insert data into TEAM, PLAYER table

```

# TOURNAMENT
insert_tournament = """
INSERT INTO Tournament (SportID, Name, StartDate, EndDate, MaxTeams) VALUES
(1,'Autumn Basketball Cup 2025','2025-09-01','2025-09-10',16),
(2,'Inter-College Volleyball 2025','2025-09-15','2025-09-22',12),
(3,'Summer Football League 2025','2025-07-01','2025-08-20',20),
(4,'Badminton Open 2025','2025-10-05','2025-10-07',32),
(5,'Table Tennis Classic 2025','2025-11-01','2025-11-03',24),
(6,'Tennis Invitational 2025','2025-11-10','2025-11-15',8),
(1,'Winter Basketball Cup 2025','2025-12-01','2025-12-10',16),
(2,'Campus Volleyball Series 2025','2025-12-12','2025-12-18',12),
(3,'Spring Soccer Cup 2025','2026-03-03','2026-03-20',20),
(4,'Spring Badminton 2025','2026-03-10','2026-03-07',32),
(5,'Summer Football 2025','2026-06-01','2026-06-20',20),
(6,'Summer Tennis 2025','2026-06-10','2026-06-15',8),
(3,'Novice Basketball Tournament','2025-07-05','2025-07-07',16),
(2,'Alumni Volleyball Cup','2025-08-01','2025-08-03',8),
(3,'Inter-department Soccer Cup','2025-08-10','2025-08-20',18),
(4,'City Badminton Challenge','2025-09-01','2025-09-03',32),
(5,'Faculty Table Tennis','2025-09-10','2025-09-12',12),
(6,'City Tennis Americas','2025-10-01','2025-10-04',16),
(1,'Regional Football 2025','2025-10-15','2025-10-20',20),
(2,'Regional Volleyball 2025','2026-02-01','2026-02-08',14),
(3,'Citywide Soccer 2026','2026-03-15','2026-03-30',24),
(4,'University Badminton League','2026-04-01','2026-04-10',40),
(5,'National Table Tennis Qualifier','2026-05-01','2026-05-07',48),
(6,'Inter-University Tennis','2026-05-10','2026-05-20',16),
(5,'Table Tennis Autumn 2025','2025-10-15','2025-10-18',20);
"""

# TEAM-TOURNAMENT
insert_team_tour = """
INSERT INTO TeamTournament (TeamID, TournamentID) VALUES
(1,1),(2,2),(3,3),(4,4),(5,5),
(6,6),(7,7),(8,8),(9,9),(10,10),
(11,11),(12,12),(13,13),(14,14),(15,15),
(16,16),(17,17),(18,18),(19,19),(20,20),
(21,21),(22,22),(23,23),(24,24),(25,25);
"""

# MATCHES
insert_match = """
INSERT INTO Match (TournamentID, TeamID, Team2ID, VenueID, RefereeID, MatchDate, Team1Score, Team2Score, WinnerTeamID) VALUES
(1,1,2,1,1,'2025-09-02 10:00',78,65,1),
(2,2,3,2,2,'2025-09-16 14:00',3,1,2),
(3,3,4,3,3,'2025-10-01 16:00',1,2,3),
(4,4,5,4,4,'2025-10-06 09:00',23,14,5),
(5,5,6,5,5,'2025-11-02 11:00',11,7,5);
"""

with engine.begin() as conn:
    conn.execute(text(insert_venue))
    conn.execute(text(insert_referee))
    conn.execute(text(insert_team))
    conn.execute(text(insert_player))
    conn.execute(text(insert_tournament))
    conn.execute(text(insert_team_tour))
    conn.execute(text(insert_match))

print("✓ Data import finished!")

# Run it
import_data()

Importing data...
✓ Data import finished!

```

Figure 6: Insert data into TOURNAMENT, TEAMTOURNAMENT, MATCH tables.

TeamID	TeamName	Department	Coach
1	Falcons	Engineering	Coach A
2	Tigers	Business	Coach B
3	Eagles	Arts	Coach C
4	Wolves	Science	Coach D
5	Panthers	Law	Coach E
6	Sharks	Medicine	Coach F
7	Rangers	Engineering	Coach G
8	Knights	Business	Coach H
9	Lions	Arts	Coach I
10	Hawks	Science	Coach J
11	Bulls	Law	Coach K
12	Wizards	Medicine	Coach L
13	Spartans	Engineering	Coach M

Figure 7: Data inserted successfully

3.4. Data Retrieval and Analytical Reports

The following are examples of SQL analytical reports generated from the database:

Query 1: For a given tournament, what is the full match schedule of each match?

SQL Statement:

```
query = """
SELECT
    m.MatchDate,
    t1.TeamName AS Team1,
    t2.TeamName AS Team2,
    v.VenueName,
    r.RefereeName,
    m.Team1Score,
    m.Team2Score,
    CASE
        WHEN m.Team1Score > m.Team2Score THEN t1.TeamName
        WHEN m.Team2Score > m.Team1Score THEN t2.TeamName
        ELSE 'Draw'
    END AS Winner
FROM `Match` m
JOIN Team t1 ON t1.TeamID = m.Team1ID
JOIN Team t2 ON t2.TeamID = m.Team2ID
JOIN Venue v ON v.VenueID = m.VenueID
JOIN Referee r ON r.RefereeID = m.RefereeID
WHERE m.TournamentID = :tid
ORDER BY m.MatchDate;
"""
```

Result:

```
User Organizer, Please select 1 option:
1. Create Tournament
2. Edit Tournament
3. Remove Tournament
4. List Tournaments
5. Schedule Match
6. View Match Schedule
7. Update Match Results
8. Department Participation Report
9. Sport Participation Report
0. Exit
=====
Your option: 6
Tournament ID: 2
(datetime.datetime(2025, 9, 16, 14, 0), 'Tigers', 'Eagles', 'Main Gym B', 'Brian Chan', 3, 1, 'Tigers')
```

Figure 8: Result of Query 1

Query 2: From each department, how many teams are participating?

SQL Statement:

```

query = """
SELECT t.Department, COUNT(*) AS Total_Teams
FROM Team t
GROUP BY t.Department
ORDER BY Total_Teams DESC;
"""

```

Result:

```

User Organizer, Please select 1 option:
1. Create Tournament
2. Edit Tournament
3. Remove Tournament
4. List Tournaments
5. Schedule Match
6. View Match Schedule
7. Update Match Results
8. Department Participation Report
9. Sport Participation Report
0. Exit
=====
Your option: 8
('Engineering', 5)
('Business', 4)
('Arts', 4)
('Science', 4)
('Law', 4)
('Medicine', 4)

```

Figure 9: Result of query 2

Query 3: For each sport, how many tournaments are organized?

SQL Statement:

```

query = """
SELECT s.SportName, COUNT(t.TournamentID) AS Total_Tournaments
FROM Sport s
LEFT JOIN Tournament t ON s.SportID = t.SportID
GROUP BY s.SportName
ORDER BY Total_Tournaments DESC;
"""

```

Result:

```
User Organizer, Please select 1 option:  
1. Create Tournament  
2. Edit Tournament  
3. Remove Tournament  
4. List Tournaments  
5. Schedule Match  
6. View Match Schedule  
7. Update Match Results  
8. Department Participation Report  
9. Sport Participation Report  
0. Exit  
=====  
Your option: 9  
('Table Tennis', 5)  
('Basketball', 4)  
('Volleyball', 4)  
('Soccer', 4)  
('Badminton', 4)  
('Tennis', 4)
```

Figure 10: Result of query 3

Query 4: How many matches had a team won and lost?

SQL Statement:

```
query = """  
SELECT  
    t.TeamID,  
    t.TeamName,  
    SUM(  
        CASE  
            WHEN m.WinnerTeamID = t.TeamID THEN 1  
            ELSE 0  
        END  
    ) AS Wins,  
    SUM(  
        CASE  
            WHEN (m.Team1ID = t.TeamID OR m.Team2ID = t.TeamID)  
                AND m.WinnerTeamID <> t.TeamID  
                AND m.WinnerTeamID IS NOT NULL  
            THEN 1  
            ELSE 0  
        END  
    ) AS Losses  
FROM Team t  
LEFT JOIN `Match` m  
ON (m.Team1ID = t.TeamID OR m.Team2ID = t.TeamID)  
WHERE t.TeamID = :team_id  
GROUP BY t.TeamID, t.TeamName;  
"""
```

Result:

```
User Team, Please select 1 option:  
1. View players  
2. View tournament  
3. View matches  
4. View performance history  
5. Roster Count  
6. Register team for tournament  
7. Add players to roster  
8. Remove player from roster  
0. Exit  
=====  
Your option: 4  
TEAM_ID: 2  
Team: Tigers; Wins 1 times; Losses 1 times.
```

Figure 11: Result of query 4

Query 5: What are the match results?

SQL Statement:

```
query = """  
SELECT  
    T.Name,  
    m.MatchDate,  
    t1.TeamName AS Team1,  
    t2.TeamName AS Team2,  
    m.Team1Score,  
    m.Team2Score,  
    w.TeamName AS Winner  
FROM `Match` m  
JOIN Tournament T ON m.TournamentID = T.TournamentID  
JOIN Team t1 ON t1.TeamID = m.Team1ID  
JOIN Team t2 ON t2.TeamID = m.Team2ID  
LEFT JOIN Team w ON w.TeamID = m.WinnerTeamID  
WHERE m.Team1Score IS NOT NULL  
    AND m.Team2Score IS NOT NULL  
ORDER BY m.MatchDate DESC;  
"""
```

Result:

```
User Spectator, Please select 1 option:  
1. View match results  
2. View upcoming matches  
0. Exit  
Your option: 1  
=====MATCH RESULTS=====  
Table Tennis Classic 2025 | 2025-11-02 11:00:00 Sharks - Panthers 11 |  
Badminton Open 2025 | 2025-10-06 09:00:00 Panthers - Wolves 21 |  
Soccer League Fall 2025 | 2025-10-02 16:00:00 Wolves - Eagles 2 |  
Inter-College Volleyball 2025 | 2025-09-16 14:00:00 Eagles - Tigers 3 |  
Autumn Basketball Cup 2025 | 2025-09-02 10:00:00 Tigers - Falcons 78 |
```

Figure 12: Result of query 5

4. Conclusion

This project successfully completed the construction of a fully functional relational database for a sport tournament registration platform, meeting the requirements. The model was designed based on transparent business rules and represented through a standardized ERD, ensuring data integrity and scalability. Implementation using mySQL, along with sample data and illustrative queries, demonstrates that the system can effectively support the needs of organizers, participating teams, and spectators. Although the project scope did not include advanced functions such as real-time interfaces, the current data provides a solid foundation for future development.