Basketball

DB MANAGEMENT SYSTEM

BY: SHAKED SHOSHAN

BASKETBALL DB SYSTEM

PURPOSE

MANAGEMENT OF THE BASKETBALL GAME DATA BASE SYSTEM. THE SYSTEM DESIGNED TO GATHER ALL THE INFORMATION RELEVANT TO BASKETBALL SPORT.

IN ADDITION THE SYSTEM CONTAINS VARIOUS PROCEDURES AND TRIGGERS THAT HELP TO MANAGE THE INFORMATION ABOUT AN INDUSTRY,

BASKETBALL DB SYSTEM

BACKGROUND

- REGISTERING NEW PLAYERS, COACHES AND REFEREES AND THEIR PESONAL DATA AND PROFESIONAL DATA. INCLUDE: NAME, CITY,COUNTRY,DOB, SALARY.

- TEAM REGISTRATION, WHICH INCLUDES INFORMATION ABOUT THE BUDGET, CITY, PLAYERS, ACHIEVEMENTS, COMPETITIONS AND FANBASE ETC.

- TRACKING THE GAME SCHEDULE, RESULTS, COMPETITIONS AND TROPHIES IN THE ENTIRE INDUSTRY

BASKETBALL DB SYSTEM

SYSTEM USERS

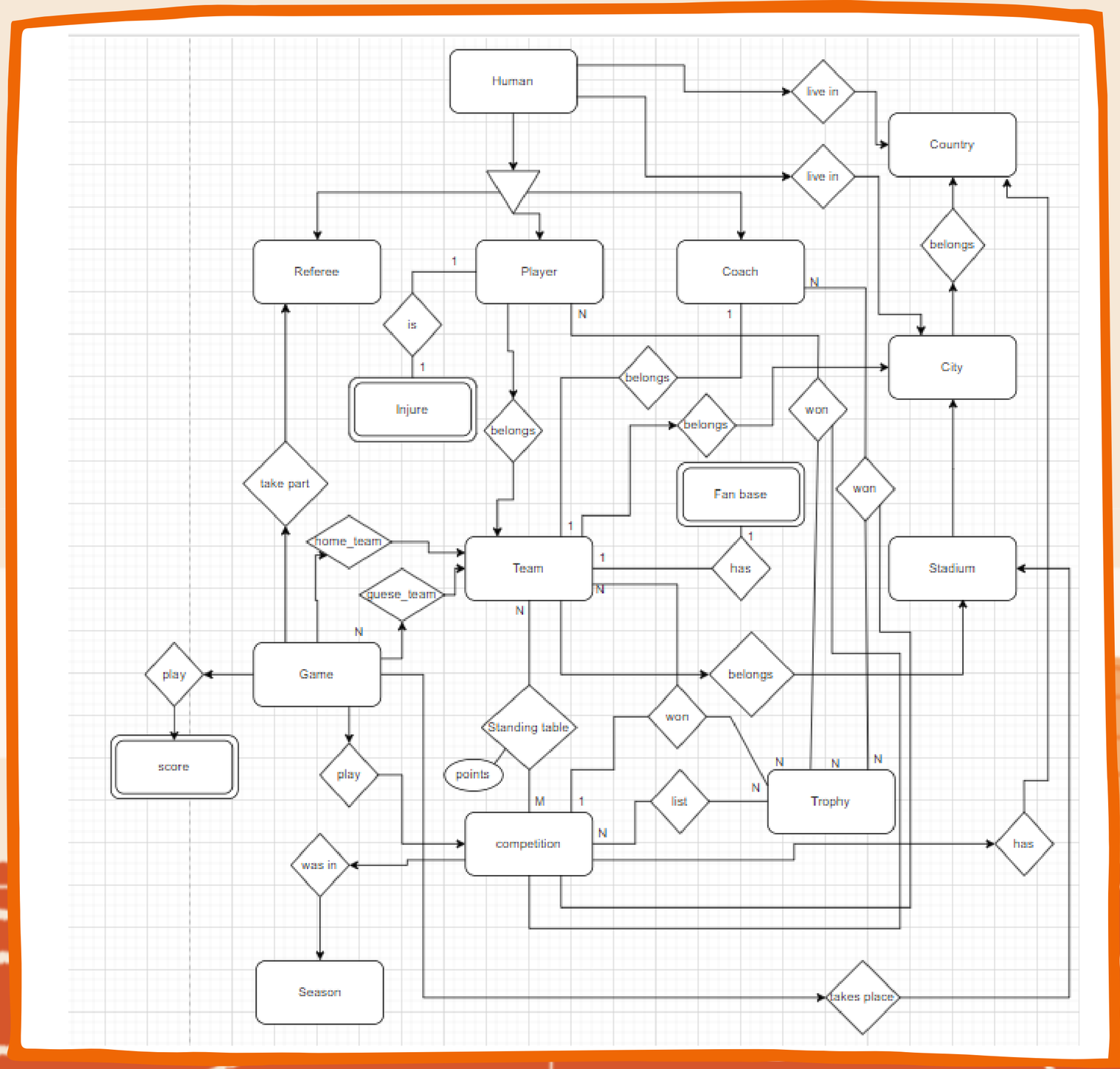
PRIVATE CUSTOMERS

CAN FOLLOW TEAMS, PLAYERS AND COMPETITIONS THEY LIKE

SYSTEM ADMINISTRATORS

WILL USE THE SYSTEM TO FOLLOW ALL THE INFORMATION THAT RUNS IN THE BASKETBALL INDUSTRY, AND UPDATE IN REAL TIME RESULTS, PLAYER TRANSITIONS, TEAM ACHIEVEMENTS AND MORE

ERD


Human
live in
Country
live in
belongs
Referee
1
Player
Coach
N
is
N
1
1
City
belongs
Injure
belongs
won
belongs
take part
Fan base
won
1
home_team
Team
1
has
Stadium
quese_team
N
N
N
play
Game
belongs
Standing table
won
score
play
points
N
N
N
M
1
list
N
Trophy
competition
N
has
was in
Season
takes place

Tables

```sql
CREATE TABLE FanBase (
    fan_count INT,
    loyalty_rating INT CHECK (loyalty_rating >= 0 AND loyalty_rating <= 10 ),
    performance_rating INT CHECK (performance_rating >= 0 AND performance_rating <= 10 ),
    avg_money_spent DECIMAL(10, 2),
    team_id INT NOT NULL,
    FOREIGN KEY (team_id) REFERENCES Teams(team_id) ON DELETE CASCADE
);

CREATE TABLE Injuries (
    injury_name VARCHAR(100),
    return_date DATE,
    player_id INT,
    FOREIGN KEY (player_id) REFERENCES Players(player_id) ON DELETE CASCADE
);


CREATE TABLE PlayerTrophy (
    player_id INT,
    trophy_id INT,
    competition_id INT NOT NULL,
    PRIMARY KEY (player_id, trophy_id,competition_id),
    FOREIGN KEY (player_id) REFERENCES Players(player_id) ON DELETE CASCADE,
    FOREIGN KEY (trophy_id) REFERENCES Trophy(trophy_id) ON DELETE CASCADE,
    FOREIGN KEY (competition_id) REFERENCES Competition(competition_id) ON DELETE CASCADE
);
```

```sql
CREATE TABLE Score (
    game_id INT NOT NULL,
    score_home INT,
    score_guest INT,
    arrived_count INT DEFAULT(NULL),
    FOREIGN KEY (game_id) REFERENCES Games(game_id) ON DELETE CASCADE
);


CREATE TABLE StandingTable (
    team_id INT,
    competition_id INT NOT NULL,
    team_points INT NOT NULL,
    FOREIGN KEY (team_id) REFERENCES Teams(team_id) ON DELETE CASCADE,
    FOREIGN KEY (competition_id) REFERENCES Competition(competition_id) ON DELETE CASCADE
);
```

```sql
CREATE TABLE CoachTrophy (
    coach_id INT,
    trophy_id INT,
    competition_id INT NOT NULL,
    PRIMARY KEY (coach_id, trophy_id,competition_id),
    FOREIGN KEY (coach_id) REFERENCES Coaches(coach_id) ON DELETE CASCADE,
    FOREIGN KEY (trophy_id) REFERENCES Trophy(trophy_id) ON DELETE CASCADE,
    FOREIGN KEY (competition_id) REFERENCES Competition(competition_id) ON DELETE CASCADE
);

CREATE TABLE TeamTrophy (
    team_id INT NOT NULL,
    trophy_id INT NOT NULL,
    competition_id INT NOT NULL,
    PRIMARY KEY (team_id, trophy_id,competition_id),
    FOREIGN KEY (team_id) REFERENCES teams(team_id) ON DELETE CASCADE,
    FOREIGN KEY (trophy_id) REFERENCES Trophy(trophy_id) ON DELETE CASCADE,
    FOREIGN KEY (competition_id) REFERENCES Competition(competition_id) ON DELETE CASCADE
);

CREATE TABLE competitionTrophy (
    competition_id INT NOT NULL,
    trophy_id INT NOT NULL,
    PRIMARY KEY (competition_id, trophy_id),
    FOREIGN KEY (trophy_id) REFERENCES Trophy(trophy_id) ON DELETE CASCADE,
    FOREIGN KEY (competition_id) REFERENCES Competition(competition_id) ON DELETE CASCADE
);
```

```sql
CREATE TABLE Human (
    human_id INT NOT NULL auto_increment PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    height DECIMAL(5, 2),
    date_of_birth DATE,
    monthly_salary INT,
    country_id INT,
    city_id INT,
    FOREIGN KEY (country_id) REFERENCES Countries(country_id),
    FOREIGN KEY (city_id) REFERENCES Cities(city_id)
);

CREATE TABLE Coaches (
    coach_id INT NOT NULL PRIMARY KEY,
    num_of_trophy INT DEFAULT(0),
    FOREIGN KEY (coach_id) REFERENCES Human(human_id) ON DELETE CASCADE
);

CREATE TABLE Teams (
    team_id INT NOT NULL auto_increment PRIMARY KEY,
    team_name VARCHAR(100),
    budget DECIMAL(25, 2),
    num_of_players INT DEFAULT(0),
    num_of_trophies INT DEFAULT(0),
    coach_id INT ,
    stadium_id INT,
    city_id INT,
    FOREIGN KEY (coach_id) REFERENCES Coaches(coach_id),
    FOREIGN KEY (stadium_id) REFERENCES Stadiums(stadium_id),
    FOREIGN KEY (city_id) REFERENCES Cities(city_id)
);
```

```sql
CREATE TABLE Season (
    season_id YEAR NOT NULL PRIMARY KEY
);

CREATE TABLE Competition (
    competition_id INT NOT NULL auto_increment PRIMARY KEY,
    competition_name VARCHAR(100),
    num_of_teams INT DEFAULT(0),
    winner_prize DECIMAL(15, 2),
    country_id INT,
    season_id YEAR NOT NULL,
    FOREIGN KEY (country_id) REFERENCES Countries(country_id),
    FOREIGN KEY (season_id) REFERENCES Season(season_id) ON DELETE CASCADE
);
```

```sql
CREATE TABLE Trophy (
    trophy_id INT NOT NULL auto_increment PRIMARY KEY,
    trophy_name VARCHAR(100),
    cost_of_manufacturing DECIMAL(10, 2),
    kind VARCHAR(10) CHECK (kind IN ('collective', 'personal'))
);

CREATE TABLE Games (
    game_id INT NOT NULL auto_increment PRIMARY KEY,
    referee_id INT DEFAULT(NULL),
    game_date DATE,
    #arrived_count INT DEFAULT(NULL),
    ticket_price DECIMAL(10, 2) DEFAULT(5.0),
    stadium_id INT NOT NULL,
    team_home_id INT NOT NULL,
    team_guese_id INT NOT NULL,
    competition_id INT,
    FOREIGN KEY (referee_id) REFERENCES Referees(referee_id),
    FOREIGN KEY (stadium_id) REFERENCES Stadiums(stadium_id),
    FOREIGN KEY (team_home_id) REFERENCES Teams(team_id),
    FOREIGN KEY (team_guese_id) REFERENCES Teams(team_id),
    FOREIGN KEY (competition_id) REFERENCES Competition(competition_id)
);
```

```sql
CREATE TABLE Players (
    player_id INT NOT NULL auto_increment PRIMARY KEY,
    human_id INT NOT NULL,
    team_id INT,
    average_points DECIMAL(4, 2) DEFAULT(0.0),
    average_assist DECIMAL(4, 2) DEFAULT(0.0),
    average_rebound DECIMAL(5, 2) DEFAULT(0.0),
    position INT CHECK (position >= 1 AND position <= 5),
    num_of_trophy INT,
    FOREIGN KEY (player_id) REFERENCES Human(human_id) ON DELETE CASCADE,
    FOREIGN KEY (team_id) REFERENCES Teams(team_id)
);

CREATE TABLE Referees (
    referee_id INT NOT NULL PRIMARY KEY,
    organization_name VARCHAR(100),
    rank_points INT DEFAULT(0),
    FOREIGN KEY (referee_id) REFERENCES Human(human_id) ON DELETE CASCADE
);

CREATE TABLE Countries (
    country_id INT NOT NULL auto_increment PRIMARY KEY,
    country_name VARCHAR(50),
    capacity INT,
    continental VARCHAR(50)
);

CREATE TABLE Cities (
    city_id INT NOT NULL auto_increment PRIMARY KEY,
    city_name VARCHAR(50),
    capacity INT,
    country_id INT,
    FOREIGN KEY (country_id) REFERENCES Countries(country_id)
);

CREATE TABLE Stadiums (
    stadium_id INT NOT NULL auto_increment PRIMARY KEY,
    city_id INT,
    street VARCHAR(50),
    street_number INT,
    seating_capacity INT,
    FOREIGN KEY (city_id) REFERENCES Cities(city_id)
```

Queries

```sql
-- List the top 3 countries with the highest total budget across all their teams:
SELECT C.country_name, SUM(T.budget) AS total_budget
FROM Countries AS C
JOIN Cities AS Ci ON C.country_id = Ci.country_id
JOIN Teams AS T ON Ci.city_id = T.city_id
GROUP BY C.country_name
ORDER BY total_budget DESC
LIMIT 3;
```

countries table

1	United
2	Brazil
3	China
4	Russia

resault

country_name	total_budget
Brazil	17248500.00
United States	15945247.00
Russia	4500000.00

team table

team_id	team_name	budget	num_of_players	num_of_trophies	coach_id	stadium_id	city_id
1	New York FC	6371197.60	27	11	1	1	1
2	Rio United	4023800.00	22	7	2	2	2
3	Beijing Dragons	7224700.00	29	11	3	2	2
4	Moscow Bears	4500000.00	27	7	4	4	4
5	Joburg Lions	3000000.00	21	3	5	5	5
6	Los Angeles Stars	4000000.00	25	6	6	6	6
7	Sao Paulo Strikers	3000000.00	19	4	7	1	1
8	Tokyo Titans	6000000.00	27	9	8	2	2
9	Berlin Thunder	4049149.40	24	7	9	1	1
10	Sydney Waves	2524900.00	17	2	10	1	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```sql
-- Find the players who have an average rebound above 8 and have won more than 2 personal trophies:
SELECT H.first_name, H.last_name
FROM Human AS H
JOIN Players AS P ON H.human_id = P.human_id
JOIN PlayerTrophy AS PT ON P.player_id = PT.player_id
JOIN Trophy AS T ON PT.trophy_id = T.trophy_id
WHERE T.kind = 'personal'
GROUP BY H.first_name, H.last_name
HAVING AVG(P.average_rebound) > 8 AND COUNT(PT.trophy_id) >= 2;
```

resault

	first_name	last_name
▶	John	Doe

human table

human_id	first_name	last_name	height	date_of_birth	monthly_salary	country_id	city_id
1	John	Doe	180.50	1990-05-15	2000	1	1
2	Maria	Silva	170.00	1985-09-22	2500	2	2
3	Li	Chen	175.20	1992-12-10	2000	3	3
4	Ivan	Petrov	190.00	1988-07-03	1000	4	4
5	Michael	Johnson	190.00	1993-05-12	3300	2	2
6	Thabo	Molefe	185.70	1995-02-28	25000	5	5
7	Sakura	Tanaka	160.00	1991-08-18	23000	10	10
8	Pedro	Rodriguez	176.40	1989-03-30	55000	2	2
9	Anna	Schmidt	170.00	1994-12-05	2000	9	9
10	Liam	Johnson	195.20	1997-06-25	8000	10	10
11	Ahmed	Ali	180.10	1990-11-12	20000	2	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

playertrophy table

player_id	trophy_id	competition_id
1	1	1
1	1	4
2	1	1
2	1	4
1	3	2
1	5	5
2	5	5
3	6	5
1	8	2
2	8	2
NULL	NULL	NULL

player table

player_id	human_id	team_id	average_points	average_assist	average_rebound	position	num_of_trophy
1	1	1	12.50	5.20	8.70	4	3
2	2	1	10.80	6.30	5.90	2	3
3	3	3	15.10	7.80	6.50	3	1
4	4	4	9.70	4.50	10.20	5	0
5	5	5	7.20	3.80	4.10	1	0
6	6	6	11.80	4.70	7.30	3	0
7	7	7	9.50	6.00	4.20	2	0
8	8	8	14.20	8.50	5.80	5	0
9	9	9	10.10	4.20	9.90	1	0
10	10	10	8.70	3.60	3.90	4	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

trophy table

trophy_id	trophy_name	cost_of_manufacturing	kind
1	Championship Cup	10000.00	collective
2	MVP Award	500.00	personal
3	Golden Boot	750.00	personal
4	Fair Play Trophy	1000.00	collective
5	League Cup	8000.00	collective
6	Championship Cup	10000.00	collective
7	Championship Cup MVP	100.00	personal
8	MVP Award	500.00	personal
9	Golden Boot	750.00	personal
10	Fair Play Trophy	1000.00	collective
NULL	NULL	NULL	NULL

```sql
-- Retrieve the cities that have at least two teams competing in a specific competition:
SELECT Ci.city_name
FROM Cities AS Ci
JOIN Teams AS T ON Ci.city_id = T.city_id
WHERE T.team_id IN (
    SELECT team_id FROM Competition
    WHERE competition_id = 1 -- Replace with the desired competition_id
)
GROUP BY Ci.city_name
HAVING COUNT(DISTINCT T.team_id) >= 2;
```

resault

city_name
New York
Rio de Janeiro

standingtable

team_id	competition_id	team_points
1	1	77
2	2	74
3	2	68
4	4	60
5	5	57
9	1	77
10	1	62

team table

team_id	team_name	budget	num_of_players	num_of_trophies	coach_id	stadium_id	city_id
1	New York FC	6371197.60	27	11	1	1	1
2	Rio United	4023800.00	22	7	2	2	2
3	Beijing Dragons	7224700.00	29	11	3	2	2
4	Moscow Bears	4500000.00	27	7	4	4	4
5	Joburg Lions	3000000.00	21	3	5	5	5
6	Los Angeles Stars	4000000.00	25	6	6	6	6
7	Sao Paulo Strikers	3000000.00	19	4	7	1	1
8	Tokyo Titans	6000000.00	27	9	8	2	2
9	Berlin Thunder	4049149.40	24	7	9	1	1
10	Sydney Waves	2524900.00	17	2	10	1	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Procedures

Team winning a competiton - insert trophy to players,team and coach, add to counter, increase fanbase and bugdet

```sql
CREATE PROCEDURE AddTeamTrophy(
    IN p_team_id INT,
    IN p_trophy_id INT,
    IN p_competition_id INT
)
BEGIN
    DECLARE p_coach_id INT;
    DECLARE winner_prize_to_add DECIMAL(25, 2);
    DECLARE current_budget DECIMAL(25, 2);
    DECLARE sum_budget DECIMAL(25, 2);


    -- insert competitionTrophy
    INSERT INTO competitionTrophy (competition_id, trophy_id)
    VALUES (p_competition_id, p_trophy_id);

    -- Insert the trophy for the team
    INSERT INTO TeamTrophy (team_id, trophy_id, competition_id)
    VALUES (p_team_id, p_trophy_id, p_competition_id);

    -- Update team's trophy count
    UPDATE Teams
    SET num_of_trophies = num_of_trophies + 1
    WHERE team_id = p_team_id;

    -- Update player's trophy count
    UPDATE Players
    SET num_of_trophy = num_of_trophy + 1
    WHERE team_id = p_team_id;
```

```sql
    -- Add the trophy for PlayerTrophy
    INSERT INTO PlayerTrophy (player_id, trophy_id, competition_id)
    SELECT player_id, p_trophy_id, p_competition_id FROM Players WHERE team_id = p_team_id;

    -- Add the trophy for CoachTrophy
    INSERT INTO CoachTrophy (coach_id, trophy_id, competition_id)
    SELECT coach_id, p_trophy_id, p_competition_id FROM Coaches
    WHERE coach_id = (SELECT coach_id FROM Teams WHERE team_id = p_team_id);


    -- Update fan_count and avg_money_spent in FanBase
    UPDATE FanBase
    SET fan_count = fan_count * 1.3, avg_money_spent = avg_money_spent * 1.1
    WHERE team_id = p_team_id;

    -- Update the budget of team after win prize
    -- get winner prize
    SELECT winner_prize INTO winner_prize_to_add
    FROM Competition
    WHERE competition_id = p_competition_id;

    SELECT budget INTO current_budget
    FROM Teams as t
    WHERE t.team_id = p_team_id;

    SET sum_budget = current_budget + winner_prize_to_add;

    UPDATE Teams as t
    SET t.budget = sum_budget
    WHERE t.team_id = p_team_id;

END //
```

```
CALL AddTeamTrophy(1, 1, 4);
CALL AddTeamTrophy(1, 5, 5);
CALL AddTeamTrophy(1, 1,1);
```

before

players table

player_id	human_id	team_id	average_points	average_assist	average_rebound	position	num_of_trophy
1	1	1	12.50	5.20	8.70	4	0
2	2	1	10.80	6.30	5.90	2	0

Teams table

team_id	team_name	budget	num_of_players	num_of_trophies	coach_id	stadium_id	city_id
1	New York FC	5099600.00	27	8	1	1	1
2	Rio United	3725000.00	22	7	2	2	2

fanbase table

fan_count	loyalty_rating	performance_rating	avg_money_spent	team_id
50000	4	5	50.00	1
70980	3	4	48.40	2

coach table

coach_id	num_of_trophy
1	3

after

players table

player_id	human_id	team_id	average_points	average_assist	average_rebound	position	num_of_trophy
1	1	1	12.50	5.20	8.70	4	3
2	2	1	10.80	6.30	5.90	2	3

Teams table

team_id	team_name	budget	num_of_players	num_of_trophies
1	New York FC	5374600.00	27	11
2	Rio United	3725000.00	22	7

fanbase table

fan_count	loyalty_rating	performance_rating	avg_money_spent	team_id
109850	4	5	66.55	1
70980	3	4	48.40	2

coach table

coach_id	num_of_trophy
1	6
2	4

Insert played game and standing table

Set a score, total people arrived and point in table

```
drop PROCEDURE if exists InsertScoreAndStanding;
DELIMITER //

CREATE PROCEDURE InsertScoreAndStanding(
    IN p_game_id INT,
    IN p_score_home INT,
    IN p_score_guest INT,
    IN p_arrived_count INT
    )
BEGIN
    DECLARE competition INT;
    DECLARE team_home INT;
    DECLARE team_guese INT;

    INSERT INTO Score (game_id, score_home, score_guest, arrived_count)
    VALUES (p_game_id, p_score_home, p_score_guest,p_arrived_count);

    -- get competition id
    SELECT competition_id INTO competition FROM Games WHERE game_id = p_game_id;

    -- get teams id
    SELECT team_home_id INTO team_home FROM Games WHERE game_id = p_game_id;
    SELECT team_guese_id INTO team_guese FROM Games WHERE game_id = p_game_id;
```

```
    -- check the winner and set points on table
    IF p_score_home > p_score_guest THEN
        -- Home team wins
        UPDATE StandingTable SET team_points = team_points + 2
        WHERE team_id = team_home AND competition_id = competition;

        UPDATE StandingTable SET team_points = team_points + 1
        WHERE team_id = team_guese AND competition_id = competition;
    ELSEIF p_score_home < p_score_guest THEN
        -- Guest team wins
        UPDATE StandingTable SET team_points = team_points + 1
        WHERE team_id = team_home AND competition_id = competition;

        UPDATE StandingTable SET team_points = team_points + 2
        WHERE team_id = team_guese AND competition_id = competition;
    END IF;
END;
```

```
CALL InsertScoreAndStanding(1,100,96,15000);
```

game_id	referee_id	game_date	ticket_price	stadium_id	team_home_id	team_guese_id	competition_id
1	2	2023-09-27	10.00	1	1	10	1

Before

Teams table

team_id	competition_id	team_points
1	1	75
2	2	69
3	2	64
4	4	60
5	5	57
9	1	75
10	1	60

Standindtable

team_id	team_name	budget
1	New York FC	5275000.00
2	Rio United	3725000.00
3	Beijing Dragons	7150000.00
4	Moscow Bears	4500000.00
5	Joburg Lions	3000000.00
6	Los Angeles Stars	4000000.00
7	Sao Paulo Strikers	3000000.00
8	Tokyo Titans	6000000.00
9	Berlin Thunder	3800000.00
10	Sydney Waves	2500000.00
NULL	NULL	NULL

Score table

game_id	score_home	score_guest	arrived_count

After

Teams table

Result Grid | Filter Rows:

team_id	competition_id	team_points
1	1	77
2	2	69
3	2	64
4	4	60
5	5	57
9	1	75
10	1	61

Standindtable

team_id	team_name	budget
1	New York FC	5374600.00
2	Rio United	3725000.00
3	Beijing Dragons	7150000.00
4	Moscow Bears	4500000.00
5	Joburg Lions	3000000.00
6	Los Angeles Stars	4000000.00
7	Sao Paulo Strikers	3000000.00
8	Tokyo Titans	6000000.00
9	Berlin Thunder	3800000.00
10	Sydney Waves	2524900.00

Score table

game_id	score_home	score_guest	arrived_count
1	100	96	15000

Transfer Player - Transfer player between teams, update players count and transfer money

```sql
-- player transfer
drop PROCEDURE if exists TransferPlayerBetweenTeams;
DELIMITER //
CREATE PROCEDURE TransferPlayerBetweenTeams(
    IN player_id_to_transfer INT,
    IN new_team_id INT,
    IN transfer_money DECIMAL(25, 2)
)
BEGIN
    DECLARE current_team_id INT;
    DECLARE old_num_of_players INT;
    DECLARE new_num_of_players INT;

    -- Get the current team of the player
    SELECT team_id INTO current_team_id
    FROM Players
    WHERE player_id = player_id_to_transfer;

    CALL TransferBudget(current_team_id, new_team_id, transfer_money);

    -- Update the number of players in the current team
    UPDATE Teams
    SET num_of_players = num_of_players - 1
    WHERE team_id = current_team_id;

    -- Update the team for the player
    UPDATE Players
    SET team_id = new_team_id
    WHERE player_id = player_id_to_transfer;

    -- Update the number of players in the new team
    SELECT num_of_players INTO old_num_of_players
    FROM Teams
    WHERE team_id = new_team_id;

    SET new_num_of_players = old_num_of_players + 1;

    UPDATE Teams
    SET num_of_players = new_num_of_players
    WHERE team_id = new_team_id;


END;
//
DELIMITER ;
```

```
CALL TransferPlayerBetweenTeams(1, 2,1);
```

Before

players table

player_id	human_id	team_id
1	1	1

teams table

team_id	team_name	budget	num_of_players
1	New York FC	5275000.00	27
2	Rio United	3725000.00	22

after

players table

	player_id	human_id	team_id
▶	1	1	2

teams table

team_id	team_name	budget	num_of_players
1	New York FC	5274999.00	26
2	Rio United	3725001.00	23

Triggers

calculate profits after games

```
DELIMITER //
CREATE TRIGGER UpdateBudgetAfterGameInsert
AFTER INSERT ON Score
FOR EACH ROW
BEGIN
    DECLARE current_ticket_price DECIMAL(10, 2);
    DECLARE current_home_team_id INT;
    DECLARE current_guese_team_id INT;
    DECLARE current_arrived_count INT;
    DECLARE budget_increase_home DECIMAL(25, 2);
    DECLARE budget_increase_guese DECIMAL(25, 2);

    -- Get the home team ID, ticket price and arrived count for the inserted game
    SELECT team_home_id, team_guese_id, ticket_price INTO current_home_team_id, current_guese_team_id, current_ticket_price
    FROM Games
    WHERE game_id = NEW.game_id;

    -- Get the ticket price  inserted game
    SELECT arrived_count INTO current_arrived_count
    FROM Score
    WHERE game_id = NEW.game_id;

    -- Calculate the budget increase based on the formula(80% go to home team 20% to guese team and 17% tax)
    SET budget_increase_home = (current_arrived_count * current_ticket_price * 0.83 * 0.8);
    SET budget_increase_guese = (current_arrived_count * current_ticket_price * 0.83 * 0.2);

    -- Update the home team's budget
    UPDATE Teams
    SET budget = budget + budget_increase_home
    WHERE team_id = current_home_team_id;

    -- Update the guese team's budget
    UPDATE Teams
    SET budget = budget + budget_increase_guese
    WHERE team_id = current_guese_team_id;

END;
//
DELIMITER ;
```

call to score

```
CALL InsertScoreAndStanding(1,100,96,15000);
```

game table

game_id	referee_id	game_date	ticket_price	stadium_id	team_home_id	team_guese_id	competition_id
1	2	2023-09-27	10.00	1	1	10	1
2	2	2023-08-27	10.00	1	1	9	1
3	2	2023-09-20	10.00	1	2	3	2
4	2	2013-09-20	10.00	1	2	3	2
5	2	2015-09-20	25.00	1	2	3	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

before

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

team_id	team_name	budget	num_of_players	num_of_trophies	coach_id	stadium_id	city_id
1	New York FC	5275000.00	27	11	1	1	1
2	Rio United	3725000.00	22	7	2	2	2
3	Beijing Dragons	7150000.00	29	11	3	2	2
4	Moscow Bears	4500000.00	27	7	4	4	4
5	Joburg Lions	3000000.00	21	3	5	5	5
6	Los Angeles Stars	4000000.00	25	6	6	6	6
7	Sao Paulo Strikers	3000000.00	19	4	7	1	1
8	Tokyo Titans	6000000.00	27	9	8	2	2
9	Berlin Thunder	3800000.00	24	7	9	1	1
10	Sydney Waves	2500000.00	17	2	10	1	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

after

team_id	team_name	budget
1	New York FC	5374600.00
2	Rio United	3725000.00
3	Beijing Dragons	7150000.00
4	Moscow Bears	4500000.00
5	Joburg Lions	3000000.00
6	Los Angeles Stars	4000000.00
7	Sao Paulo Strikers	3000000.00
8	Tokyo Titans	6000000.00
9	Berlin Thunder	3800000.00
10	Sydney Waves	2524900.00
NULL	NULL	NULL

check country consistency -

check if team in competition on same country, if country in competition is null so it is global competition

```
CREATE TRIGGER check_country_consistency
AFTER INSERT ON StandingTable
FOR EACH ROW
BEGIN
    DECLARE competition_country_id INT;
    DECLARE team_country_id INT;

    -- Get the country_id of the competition
    SELECT country_id INTO competition_country_id
    FROM Competition
    WHERE competition_id = NEW.competition_id;

    -- Check if there's a competition_country_id and the team's country_id is not NULL
    IF competition_country_id IS NOT NULL AND NEW.team_id IS NOT NULL THEN

        -- Get the country_id of the team's city
        SELECT c.country_id INTO team_country_id
        FROM Cities c
        JOIN Teams t ON c.city_id = t.city_id
        WHERE t.team_id = NEW.team_id;

        -- Check if the team's country_id is different from the competition's country_id
        IF team_country_id <> competition_country_id THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Team and competition are from different countries';
        END IF;
    END IF;

END;
```

```sql
INSERT INTO StandingTable (team_id, competition_id, team_points) # legal
VALUES (1, 1, 75);
INSERT INTO StandingTable (team_id, competition_id, team_points) #illegal
VALUES (2, 1, 69);
INSERT INTO StandingTable (team_id, competition_id, team_points) #llegal
VALUES (2, 3, 69);
```

team_id	team_name	budget	num_of_players	num_of_trophies	coach_id	stadium_id	city_id
1	New York FC	5275000.00	26	11	1	1	1
2	Rio United	3725000.00	23	7	2	2	2

competition_id	competition_name	num_of_teams	winner_prize	country_id
1	National Championship	10	50000.00	1
2	International Cup	16	100000.00	2
3	Regional Tournament	8	25000.00	NULL

city_id	city_name	capacity	country_id
1	New York	8419600	1
2	Rio de Janeiro	6718903	2

✓	17842	15:53:19	INSERT INTO StandingTable (team_id, competition_id, team_points) VALUES (1, 1, 75)	1 row(s) affected
✗	17843	15:53:19	INSERT INTO StandingTable (team_id, competition_id, team_points) VALUES (2, 1, 69)	Error Code: 1644. Team and competition are from different countries
✓	17845	16:03:05	INSERT INTO StandingTable (team_id, competition_id, team_points) #illegal VALUES (2, 3, 69)	1 row(s) affected

check game availability-

Check if the team and the referee are free on the same day when the new game is scheduled

```
CREATE TRIGGER check_game_availability
BEFORE INSERT ON Games
FOR EACH ROW
BEGIN
-- check for no duplicate game
    IF EXISTS (
        SELECT 1
        FROM Games
        WHERE game_date = NEW.game_date
            AND (team_home_id = NEW.team_home_id OR team_guese_id = NEW.team_guese_id)
    ) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'One of the teams already has a game on the same date.';
    END IF;

-- check the available of the refeere
    IF EXISTS (
        SELECT 1
        FROM Games
        WHERE game_date = NEW.game_date AND referee_id = NEW.referee_id
    ) THEN
        -- The referee already has another game on the same date, prevent insertion
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'The referee already has another game on the same date.';
    END IF;
END;
//

DELIMITER ;
```

```sql
INSERT INTO Games (referee_id, game_date, stadium_id, team_home_id, team_guese_id, competition_id)
VALUES ( 2, '2023-06-17', 1, 10, 9, 1); # legall

INSERT INTO Games (referee_id, game_date, stadium_id, team_home_id, team_guese_id, competition_id)
VALUES ( 1, '2023-06-17', 1, 10, 1, 1); # not available team (10)

INSERT INTO Games (referee_id, game_date, stadium_id, team_home_id, team_guese_id, competition_id)
VALUES ( 2, '2023-06-17', 1, 7, 1, 1); # not available refeere
```

18275	16:40:07	INSERT INTO Games (referee_id, game_date, stadium_id, team_home_id, team_guese_id, competition_id)	VALUES (2, '2023-06-17', 1, 10, 9, 1)	1 row(s) affected
18276	16:40:07	INSERT INTO Games (referee_id, game_date, stadium_id, team_home_id, team_guese_id, competition_id)	VALUES (1, '2023-06-17', 1, 10, 9, 1)	Error Code: 1644. One of the teams already has a game on the same date.
18277	16:40:13	INSERT INTO Games (referee_id, game_date, stadium_id, team_home_id, team_guese_id, competition_id)	VALUES (2, '2023-06-17', 1, 7, 1, 1)	Error Code: 1644. The referee already has another game on the same date.

Questions?