DBMS REPORT Cricket Database

SRN1: PES1UG22CS338

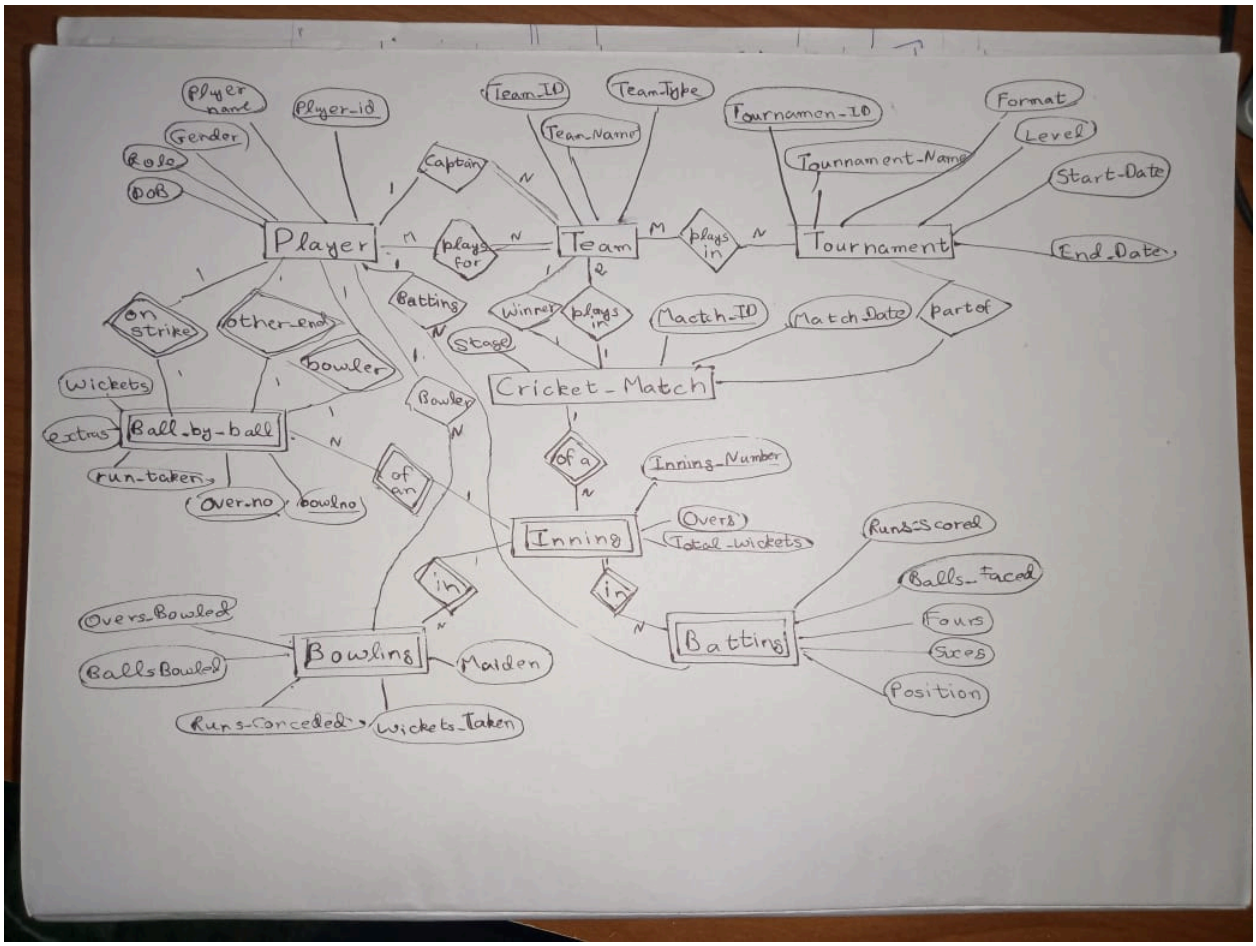
NAME1: Mayank Sharma

SRN2: PES1UG22CS364

NAME2:Munis Shafiq

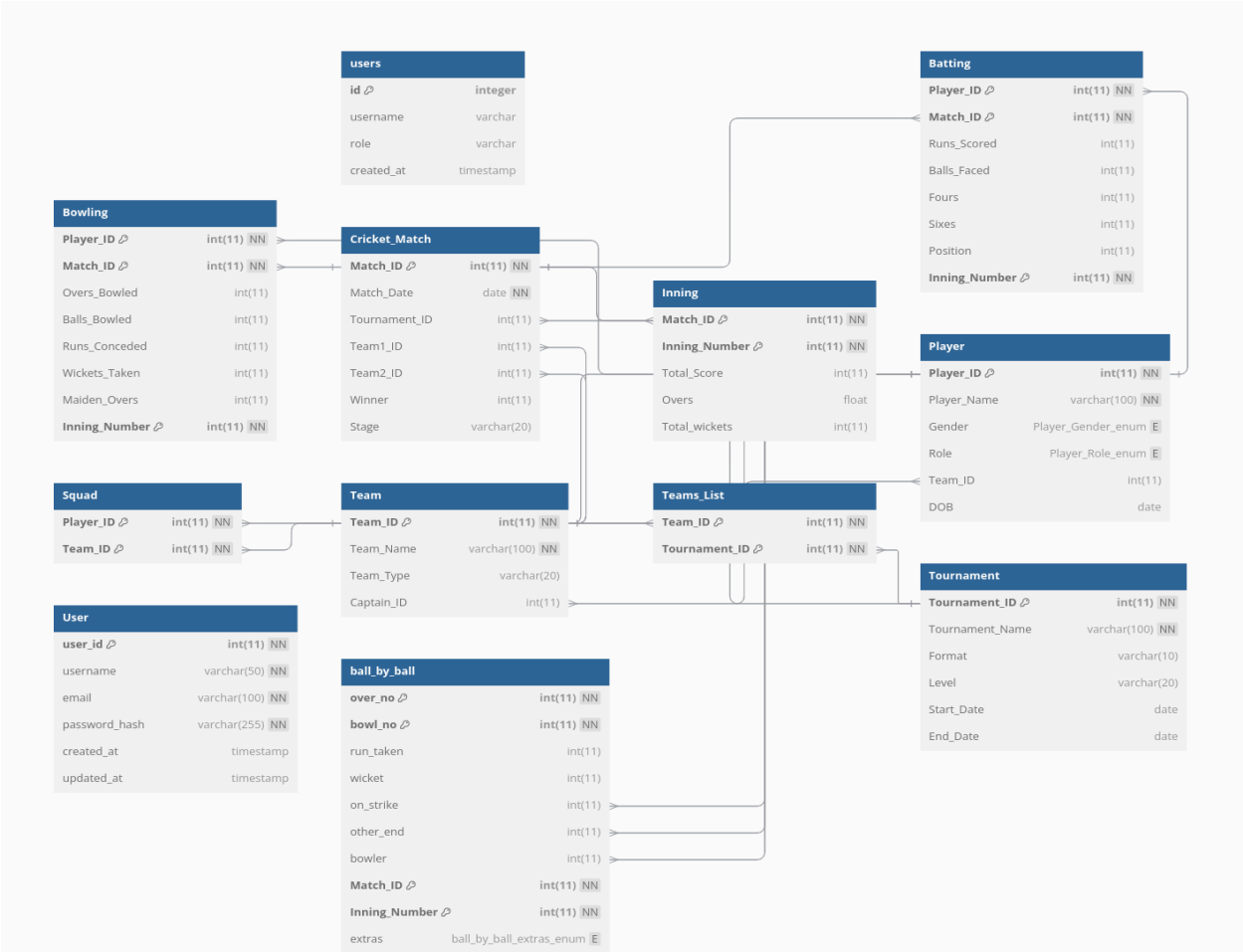
Github Link: [CricketTeamSelector]

ER-Diagram:



DBMS REPORT Cricket Database

Relational Schema:



Views :

1. ***Player Batting Stats:***

```
CREATE OR REPLACE VIEW Player_Batting_Stats AS
SELECT
        b.Player_ID,
        p.Player_Name,
        COUNT(DISTINCT b.Match_ID) AS Matches_Played,
        SUM(b.Runs_Scored) AS Total_Runs,
        SUM(b.Balls_Faced) AS Total_Balls,
        SUM(b.Fours) AS Total_Fours,
        SUM(b.Sixes) AS Total_Sixes,
        AVG(b.Runs_Scored) AS Average_Runs_Scored
FROM
        Batting b
JOIN
```

```
        Player p ON b.Player_ID = p.Player_ID
GROUP BY
        b.Player_ID, p.Player_Name;
```

2. ***Player Bowling Stats:***

```
CREATE OR REPLACE VIEW Player_Bowling_Stats AS
SELECT
        b.Player_ID,
        p.Player_Name,
        COUNT(DISTINCT b.Match_ID) AS Matches_Played,
        SUM(b.Overs_Bowled) AS Total_Overs_Bowled,
        SUM(b.Balls_Bowled) AS Total_Balls_Bowled,
        SUM(b.Runs_Conceded) AS Total_Runs_Conceded,
        SUM(b.Wickets_Taken) AS Total_Wickets_Taken,
        SUM(b.Maiden_Overs) AS Total_Maiden_Overs
FROM
        Bowling b
JOIN
        Player p ON b.Player_ID = p.Player_ID
GROUP BY
        b.Player_ID, p.Player_Name;
```

Procedure :

```
DELIMITER //
CREATE PROCEDURE GetPlayerByID(IN p_id INT)
BEGIN
        SELECT * FROM Player WHERE Player_ID = p_id;
END //
DELIMITER ;

DELIMITER //
CREATE PROCEDURE InsertPlayer(
        IN p_name VARCHAR(100),
        IN p_gender CHAR(1),
        IN p_role VARCHAR(50),
        IN p_team_id INT,
        IN p_dob DATE
)
BEGIN
        INSERT INTO Player (Player_Name, Gender, Role, Team_ID, DOB)
        VALUES (p_name, p_gender, p_role, p_team_id, p_dob);
```

```
END //
DELIMITER ;
```

Triggers Used:

```
DELIMITER $$

CREATE TRIGGER after_ball_insert
AFTER INSERT ON ball_by_ball
FOR EACH ROW
BEGIN
        DECLARE total_runs INT DEFAULT 0;
        DECLARE total_wickets INT DEFAULT 0;
        DECLARE total_balls INT DEFAULT 0;

        INSERT INTO Inning (Match_ID, Inning_Number, Total_Score, Total_Wickets, Overs)
        VALUES (NEW.Match_ID, NEW.Inning_Number, 0, 0, 0)
        ON DUPLICATE KEY UPDATE
        Total_Score = Total_Score;

        INSERT INTO Batting (Match_ID, Inning_Number, Player_ID, Runs_Scored,
Balls_Faced, Fours, Sixes)
        VALUES (NEW.Match_ID, NEW.Inning_Number, NEW.on_strike, 0, 0, 0, 0)
        ON DUPLICATE KEY UPDATE
        Runs_Scored = Runs_Scored;

        INSERT INTO Bowling (Match_ID, Inning_Number, Player_ID, Overs_Bowled,
Balls_Bowled, Runs_Conceded, Wickets_Taken)
        VALUES (NEW.Match_ID, NEW.Inning_Number, NEW.bowler, 0, 0, 0, 0)
        ON DUPLICATE KEY UPDATE
        Runs_Conceded = Runs_Conceded;

        SELECT SUM(run_taken), SUM(wicket), COUNT(*) INTO total_runs, total_wickets,
total_balls
        FROM ball_by_ball
        WHERE Match_ID = NEW.Match_ID AND Inning_Number = NEW.Inning_Number;

        UPDATE Inning
        SET Total_Score = total_runs,
        Total_Wickets = total_wickets,
        Overs = FLOOR(total_balls / 6) + (total_balls % 6) / 6.0
        WHERE Match_ID = NEW.Match_ID AND Inning_Number = NEW.Inning_Number;
```

```
END$$

DELIMITER ;
DELIMITER $$

-- Trigger to ensure Player's DOB is valid (10 years or older)
CREATE TRIGGER before_player_insert_dob_check
BEFORE INSERT ON Player
FOR EACH ROW
BEGIN
        IF NEW.DOB > DATE_SUB(CURDATE(), INTERVAL 10 YEAR) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Player must be at least 10 years old.';
        END IF;
END$$

CREATE TRIGGER before_player_update_dob_check
BEFORE UPDATE ON Player
FOR EACH ROW
BEGIN
        IF NEW.DOB > DATE_SUB(CURDATE(), INTERVAL 10 YEAR) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Player must be at least 10 years old.';
        END IF;
END$$

-- Trigger to ensure Gender is valid ENUM('M', 'F', 'O')
CREATE TRIGGER before_player_insert_gender_check
BEFORE INSERT ON Player
FOR EACH ROW
BEGIN
        IF NEW.Gender NOT IN ('M', 'F', 'O') THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Gender must be one of M, F, or O.';
        END IF;
END$$

CREATE TRIGGER before_player_update_gender_check
BEFORE UPDATE ON Player
FOR EACH ROW
BEGIN
        IF NEW.Gender NOT IN ('M', 'F', 'O') THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Gender must be one of M, F, or O.';
```

```
    END IF;
END$$

-- Trigger to ensure Role is valid
CREATE TRIGGER before_player_insert_role_check
BEFORE INSERT ON Player
FOR EACH ROW
BEGIN
    IF NEW.Role NOT IN ('Batsman', 'Bowler', 'All-Rounder', 'Wicket-Keeper') THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Role must be one of Batsman, Bowler, All-Rounder, or
Wicket-Keeper.';
    END IF;
END$$

CREATE TRIGGER before_player_update_role_check
BEFORE UPDATE ON Player
FOR EACH ROW
BEGIN
    IF NEW.Role NOT IN ('Batsman', 'Bowler', 'All-Rounder', 'Wicket-Keeper') THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Role must be one of Batsman, Bowler, All-Rounder, or
Wicket-Keeper.';
    END IF;
END$$

-- Trigger to ensure Team_ID exists in Team table
CREATE TRIGGER before_player_insert_team_check
BEFORE INSERT ON Player
FOR EACH ROW
BEGIN
    IF NOT EXISTS (SELECT 1 FROM Team WHERE Team_ID = NEW.Team_ID) THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Team_ID does not exist in the Team table.';
    END IF;
END$$

CREATE TRIGGER before_player_update_team_check
BEFORE UPDATE ON Player
FOR EACH ROW
BEGIN
    IF NOT EXISTS (SELECT 1 FROM Team WHERE Team_ID = NEW.Team_ID) THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Team_ID does not exist in the Team table.';
```

```
    END IF;
END$$

DELIMITER ;
```

Ball-by-Ball Queries:

1. **Select All Ball-by-Ball Stats**
   ```
   SELECT * FROM ball_by_ball
   ```
2. **Insert New Ball-by-Ball Stats**
   ```
   INSERT INTO ball_by_ball (over_no, bowl_no, run_taken, wicket, on_strike, other_end, bowler, Match_ID, Inning_Number) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)
   ```
3. **Update Ball-by-Ball Stats**
   ```
   UPDATE ball_by_ball SET run_taken = COALESCE(%s, run_taken), wicket = COALESCE(%s, wicket), on_strike = COALESCE(%s, on_strike), other_end = COALESCE(%s, other_end), bowler = COALESCE(%s, bowler) WHERE Match_ID = %s AND Inning_Number = %s AND over_no = %s AND bowl_no = %s
   ```
4. **Delete Ball-by-Ball Stats**
   ```
   DELETE FROM ball_by_ball WHERE Match_ID = %s AND Inning_Number = %s AND over_no = %s AND bowl_no = %s
   ```

Batting Queries:

1. **Select All Batting Stats**
   ```
   SELECT Batting.*, Player.Player_Name FROM Batting JOIN Player ON Batting.Player_ID = Player.Player_ID
   ```
2. **Select Batting Stats by Player**
   ```
   SELECT Batting.*, Player.Player_Name FROM Batting JOIN Player ON Batting.Player_ID = Player.Player_ID WHERE Batting.Player_ID = %s
   ```
3. **Select Batting Scorecard for a Match and Inning**
   ```
   SELECT Batting.Player_ID, Player.Player_Name, Batting.Runs_Scored, Batting.Balls_Faced, Batting.Fours, Batting.Sixes, Batting.Position FROM Batting JOIN Player ON Batting.Player_ID = Player.Player_ID WHERE Batting.Match_ID = %s AND Batting.Inning_Number = %s
   ```

4. **Select Batting Stats by Player and Match ID**
```
SELECT Batting.*, Player.Player_Name FROM Batting JOIN Player ON Batting.Player_ID = Player.Player_ID WHERE Batting.Player_ID = %s AND Batting.Match_ID = %s
```
5. **Insert New Batting Stats**
```
INSERT INTO Batting (Player_ID, Match_ID, Runs_Scored, Balls_Faced, Fours, Sixes, Position, Inning_Number) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
```
6. **Update Batting Stats**
```
UPDATE Batting SET Runs_Scored = COALESCE(%s, Runs_Scored), Balls_Faced = COALESCE(%s, Balls_Faced), Fours = COALESCE(%s, Fours), Sixes = COALESCE(%s, Sixes), Position = COALESCE(%s, Position), Inning_Number = COALESCE(%s, Inning_Number) WHERE Player_ID = %s AND Match_ID = %s AND Inning_Number = %s
```
7. **Delete Batting Stats**
```
DELETE FROM Batting WHERE Player_ID = %s AND Match_ID = %s AND Inning_Number = %s
```

Bowling Queries:

1. **Select All Bowling Stats**
```
SELECT Bowling.*, Player.Player_Name FROM Bowling JOIN Player ON Bowling.Player_ID = Player.Player_ID
```
2. **Select Bowling Scorecard for a Match and Inning**
```
SELECT Bowling.Player_ID, Player.Player_Name, Bowling.Overs_Bowled, Bowling.Balls_Bowled, Bowling.Runs_Conceded, Bowling.Wickets_Taken FROM Bowling JOIN Player ON Bowling.Player_ID = Player.Player_ID WHERE Bowling.Match_ID = %s AND Bowling.Inning_Number = %s
```
3. **Select Bowling Stats by Player for a Specific Match**
```
SELECT Bowling.*, Player.Player_Name FROM Bowling JOIN Player ON Bowling.Player_ID = Player.Player_ID WHERE Bowling.Match_ID = %s AND Bowling.Player_ID = %s
```
4. **Insert New Bowling Stats**
```
INSERT INTO Bowling (Player_ID, Match_ID, Inning_Number, Overs_Bowled, Balls_Bowled, Runs_Conceded, Wickets_Taken, Maiden_Overs) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
```

5. **Update Bowling Stats**
   ```
   UPDATE Bowling SET Overs_Bowled = COALESCE(%s, Overs_Bowled), Balls_Bowled = COALESCE(%s, Balls_Bowled), Runs_Conceded = COALESCE(%s, Runs_Conceded), Wickets_Taken = COALESCE(%s, Wickets_Taken), Maiden_Overs = COALESCE(%s, Maiden_Overs) WHERE Match_ID = %s AND Player_ID = %s AND Inning_Number = %s
   ```
6. **Delete Bowling Stats**
   ```
   DELETE FROM Bowling WHERE Match_ID = %s AND Player_ID = %s AND Inning_Number = %s
   ```

CricketMatch Queries:

1. **Get All Matches**
   SELECT
   cm.Match_ID,
   cm.Match_Date,
   cm.Tournament_ID,
   cm.Team1_ID,
   cm.Team2_ID,
   cm.Winner,
   cm.Stage,
   (SELECT Team_Name FROM Team WHERE Team_ID = cm.Team1_ID) AS Team1_Name,
   (SELECT Team_Name FROM Team WHERE Team_ID = cm.Team2_ID) AS Team2_Name,
   (SELECT Team_Name FROM Team WHERE Team_ID = cm.Winner) AS Winner_Name
   FROM Cricket_Match cm;
2. **Get Match by ID**
   SELECT
   cm.Match_ID,
   cm.Match_Date,
   cm.Tournament_ID,
   cm.Team1_ID,
   cm.Team2_ID,
   cm.Winner,
   cm.Stage,
   (SELECT Team_Name FROM Team WHERE Team_ID = cm.Team1_ID) AS Team1_Name,
   (SELECT Team_Name FROM Team WHERE Team_ID = cm.Team2_ID) AS Team2_Name,

```
(SELECT Team_Name FROM Team WHERE Team_ID = cm.Winner) AS Winner_Name
FROM Cricket_Match cm
WHERE cm.Match_ID = %s;
```

3. **Add New Match**
```
INSERT INTO Cricket_Match (Match_Date, Tournament_ID, Team1_ID, Team2_ID,
Winner, Stage)
VALUES (%s, %s, %s, %s, %s, %s);
```
4. **Update Match**
```
UPDATE Cricket_Match
SET
Match_Date = COALESCE(%s, Match_Date),
Tournament_ID = COALESCE(%s, Tournament_ID),
Team1_ID = COALESCE(%s, Team1_ID),
Team2_ID = COALESCE(%s, Team2_ID),
Winner = COALESCE(%s, Winner),
Stage = COALESCE(%s, Stage)
WHERE Match_ID = %s;
```
5. **Delete Match**
```
DELETE FROM Cricket_Match
WHERE Match_ID = %s;
```

Inning Queries:

1. **Get All Innings**
```
SELECT * FROM Inning;
```
2. **Get Inning Scorecard**
```
SELECT Total_Score, Overs, Total_Wickets
FROM Inning
WHERE Match_ID = %s AND Inning_Number = %s;
```
3. **Add New Inning**
```
INSERT INTO Inning (Match_ID, Inning_Number, Total_Score, Overs, Total_Wickets)
VALUES (%s, %s, %s, %s, %s);
```
4. **Update Inning**
```
UPDATE Inning
SET Total_Score = COALESCE(%s, Total_Score),
Overs = COALESCE(%s, Overs),
Total_Wickets = COALESCE(%s, Total_Wickets)
WHERE Match_ID = %s AND Inning_Number = %s;
```
5. **Delete Inning**
```
DELETE FROM Inning
WHERE Match_ID = %s AND Inning_Number = %s;
```

DBMS REPORT Cricket Database

Player Queries:

1. **Get All Players**
   SELECT * FROM Player;
2. **Get Player By ID**
   CALL GetPlayerByID(%s);
3. **Add Player**
   CALL InsertPlayer(%s, %s, %s, %s, %s);
4. **Update Player**
   UPDATE Player
   SET Player_Name = %s, Gender = %s, Role = %s, Team_ID = %s, DOB = %s
   WHERE Player_ID = %s;
5. **Delete Player By ID**
   DELETE FROM Player
   WHERE Player_ID = %s;

Player Batting Stats Queries:

1. **Get All Player Batting Stats**
   SELECT * FROM Player_Batting_Stats;
2. **Get Player Batting Stats By Player ID**
   SELECT * FROM Player_Batting_Stats WHERE Player_ID = %s;

Player Bowling Stats Queries:

1. **Get All Player Bowling Stats**
   SELECT * FROM Player_Bowling_Stats;
2. **Get Player Bowling Stats By Player ID**
   SELECT * FROM Player_Bowling_Stats WHERE Player_ID = %s;

Squad Queries:

1. **Get all squad entries with player and team names:**

```
SELECT
s.Player_ID,
s.Team_ID,
p.Player_Name,
t.Team_Name
FROM Squad s
LEFT JOIN Player p ON s.Player_ID = p.Player_ID
LEFT JOIN Team t ON s.Team_ID = t.Team_ID
```

2. **Get all players in a specific team:**

```
SELECT
s.Player_ID,
s.Team_ID,
p.Player_Name,
t.Team_Name
FROM Squad s
LEFT JOIN Player p ON s.Player_ID = p.Player_ID
LEFT JOIN Team t ON s.Team_ID = t.Team_ID
WHERE s.Team_ID = %s
```

3. **Add a new squad entry:**

```
INSERT INTO Squad (Player_ID, Team_ID)
VALUES (%s, %s)
```

4. **Get a specific squad entry by player and team IDs:**

```
SELECT
s.Player_ID,
s.Team_ID,
p.Player_Name,
t.Team_Name
FROM Squad s
LEFT JOIN Player p ON s.Player_ID = p.Player_ID
LEFT JOIN Team t ON s.Team_ID = t.Team_ID
WHERE s.Player_ID = %s AND s.Team_ID = %s
```

5. **Update an existing squad entry:**

```
UPDATE Squad
SET Player_ID = COALESCE(%s, Player_ID),
Team_ID = COALESCE(%s, Team_ID)
WHERE Player_ID = %s AND Team_ID = %s
```

6. **Delete a specific squad entry:**

```
DELETE FROM Squad WHERE Player_ID = %s AND Team_ID = %s
```

Team Queries:

1. **Get all teams:**

```
SELECT * FROM Team
```

2. **Get a team by its ID:**

```
SELECT * FROM Team WHERE Team_ID = %s
```

3. **Add a new team:**

```
INSERT INTO Team (Team_Name, Team_Type, Captain_ID)
VALUES (%s, %s, %s)
```

4. **Update an existing team:**

```
UPDATE Team SET Team_Name = %s, Team_Type = %s, Captain_ID = %s
WHERE Team_ID = %s
```

5. **Delete a team by its ID:**

```
DELETE FROM Team WHERE Team_ID = %s
```

Team-List Queries:

Fetch all teams from the Teams_List table: SELECT * FROM Teams_List
Add a new team entry to the Teams_List table: INSERT INTO Teams_List (Team_ID, Tournament_ID)
VALUES (%s, %s)
Fetch a specific team entry by team and tournament IDs: SELECT * FROM Teams_List
WHERE Team_ID = %s AND Tournament_ID = %s
Update an existing team entry in the Teams_List table: UPDATE Teams_List
SET Team_ID = COALESCE(%s, Team_ID),
Tournament_ID = COALESCE(%s, Tournament_ID)
WHERE Team_ID = %s AND Tournament_ID = %s
Delete a specific team entry by team and tournament IDs: DELETE FROM Teams_List
WHERE Team_ID = %s AND Tournament_ID = %s

Tournament Queries:

Retrieve all tournaments from the database: SELECT * FROM Tournament
Retrieve a tournament by its ID: SELECT * FROM Tournament WHERE Tournament_ID = %s
Add a new tournament to the database: INSERT INTO Tournament (Tournament_Name, Format, Level, Start_Date, End_Date)
VALUES (%s, %s, %s, %s, %s)
Update the tournament in the database: UPDATE Tournament
SET Tournament_Name = %s, Format = %s, Level = %s, Start_Date = %s, End_Date = %s
WHERE Tournament_ID = %s
Delete a tournament from the database: DELETE FROM Tournament WHERE Tournament_ID = %s

User Queries:

Add a new user to the database: INSERT INTO User (username, email, password_hash) VALUES (%s, %s, %s)
Retrieve all users from the database: SELECT * FROM User
Retrieve a user by their ID: SELECT * FROM User WHERE user_id = %s
Update a user's details in the database: UPDATE User
SET username = %s, email = %s, password_hash = %s, updated_at = %s
WHERE user_id = %s
Delete a user from the database: DELETE FROM User WHERE user_id = %s