IPL Management System

ARYAK LALWANI
AAKASH ARORA

INDEX

Introduction

Our project endeavors to create a comprehensive IPL (Indian Premier League) Management System, designed to meticulously capture, manage, and analyze the multifaceted data landscape of this premier professional 20-20 cricket league. The IPL, renowned for its global talent pool and intense franchise-based competition, involves far more than just the initial player auction. Our system aims to model the entirety of the IPL ecosystem, from pre-season preparations like auctions and player retentions, through the dynamic progression of the tournament including match schedules, live scores, and points table updates, to the rich tapestry of historical records and player statistics that define the league's legacy.

At the core of our system lies a sophisticated relational database, architected to store and manage an extensive array of data categories: detailed player profiles, franchise team specifics, owner consortiums, auction outcomes, match fixtures and results, individual player performance statistics per match, season-specific leaderboards (like Orange and Purple Cap contenders), current team standings, official league announcements, venue details, and deep historical archives of past IPL seasons including championship wins and all-time player records.

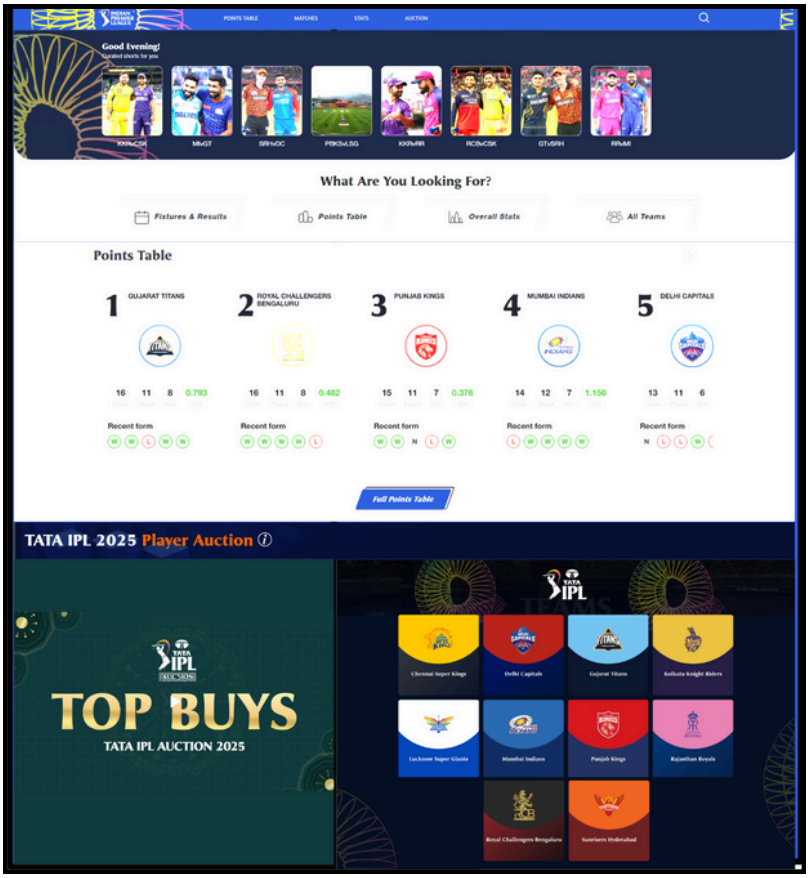
Key entities such as Players, Teams, Seasons, and Matches feature unique, auto-incremented identifiers (e.g., player_id, team_id, match_id) to ensure distinctness and facilitate robust data tracking and relationship management. The Players table, for instance, will not only store basic information like name and nationality but also their playing role, auction status (sold, unsold, retained), contract details for each season including salary, and links to their detailed performance metrics. Similarly, the Teams table will house information beyond just name and captain, including details of their ownership group, historical performance statistics, post-auction funds remaining, and current squad composition (total and overseas player counts).

A critical aspect of the system is its ability to model complex relationships through foreign key constraints, ensuring data integrity and consistency. For example, a PlayerContracts table will link players to specific teams for a given season, detailing their salary and status. Match results will dynamically update a PointsTable for the current season, reflecting wins, losses, and Net Run Rate, while also feeding into individual PlayerMatchStats. The system will also manage an AuctionLog to track base prices, sold prices, and the status of every player entering the auction pool for each season.

Beyond current season management, dedicated tables will store TeamHistoricalStats (overall win/loss records, highest/lowest totals for each franchise)

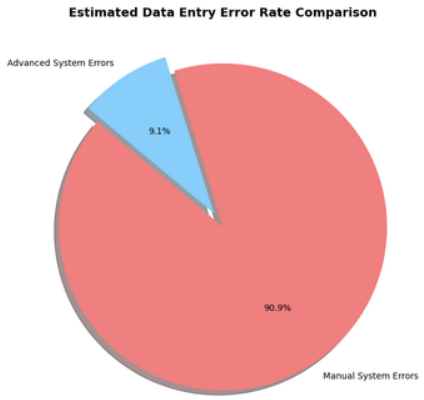
and LeagueHistoricalRecords (all-time leading run-scorers, wicket-takers, best averages, etc.), providing a rich repository for analytical insights and fan engagement. Information regarding team owners, their associated companies, and purchase years adds another dimension to the league's operational context. Furthermore, a NewsAnnouncements module will capture official communications, and a Venues table will manage stadium information used across seasons.

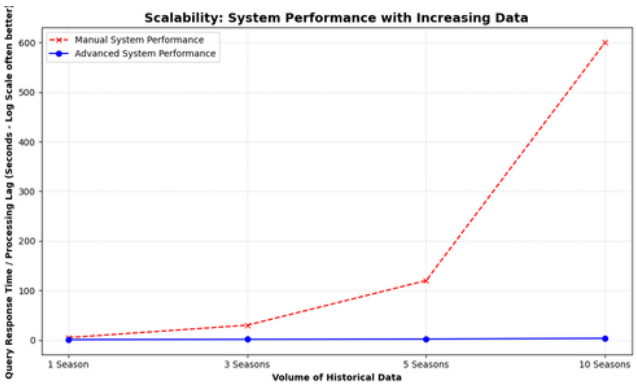
Secure access is paramount, managed through an Admins table storing credentials and potentially roles for differentiated system access, ensuring that data entry and system management are performed by authorized personnel.
Overall, this IPL Management System is designed to be a holistic and authoritative platform. By creating a detailed and interconnected database schema, the project aims to track every significant aspect of the IPL—from player acquisition and team formation, through the excitement of match-day action and season progression, to the preservation and analysis of its rich history. The implementation of robust data integrity measures, coupled with the potential for advanced querying and procedural logic, will ensure that the system serves as an accurate, consistent, reliable, and invaluable tool for anyone involved with or following the Indian Premier League.



System vs. Manual: IPL Data Management Impact

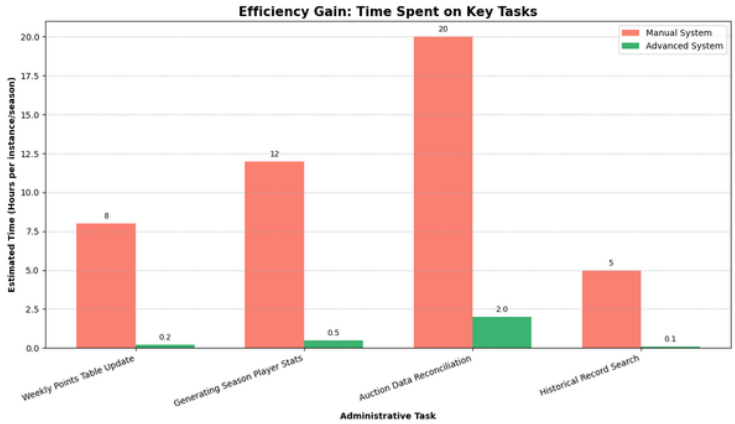
1. Data Integrity & Accuracy:
   - Manual: Prone to typos, inconsistencies, and outdated entries; corrections are unreliable and slow.
   - System: Enforces rules, constraints, and automated calculations, minimizing human error for trustworthy data.
   - Impact: Reliable data foundation for decisions, reporting, and engagement; less cleaning needed.





2. Efficiency and Speed:
   - Manual: Slow, labor-intensive data entry for matches, stats, and auctions; reports require manual compilation.
   - System: Automates capture, calculations, and updates; instant reports via queries reduce administrative load.
   - Impact: Frees personnel for strategic tasks; provides real-time information availability for all.

3. Scalability & Data Volume:
   - Manual: Struggles with growing data volumes, becoming error-prone and difficult to manage effectively.
   - System: Efficiently handles vast data, allowing seamless expansion for new seasons or categories.
   - Impact: System grows with the league, ensuring long-term performance and data reliability.

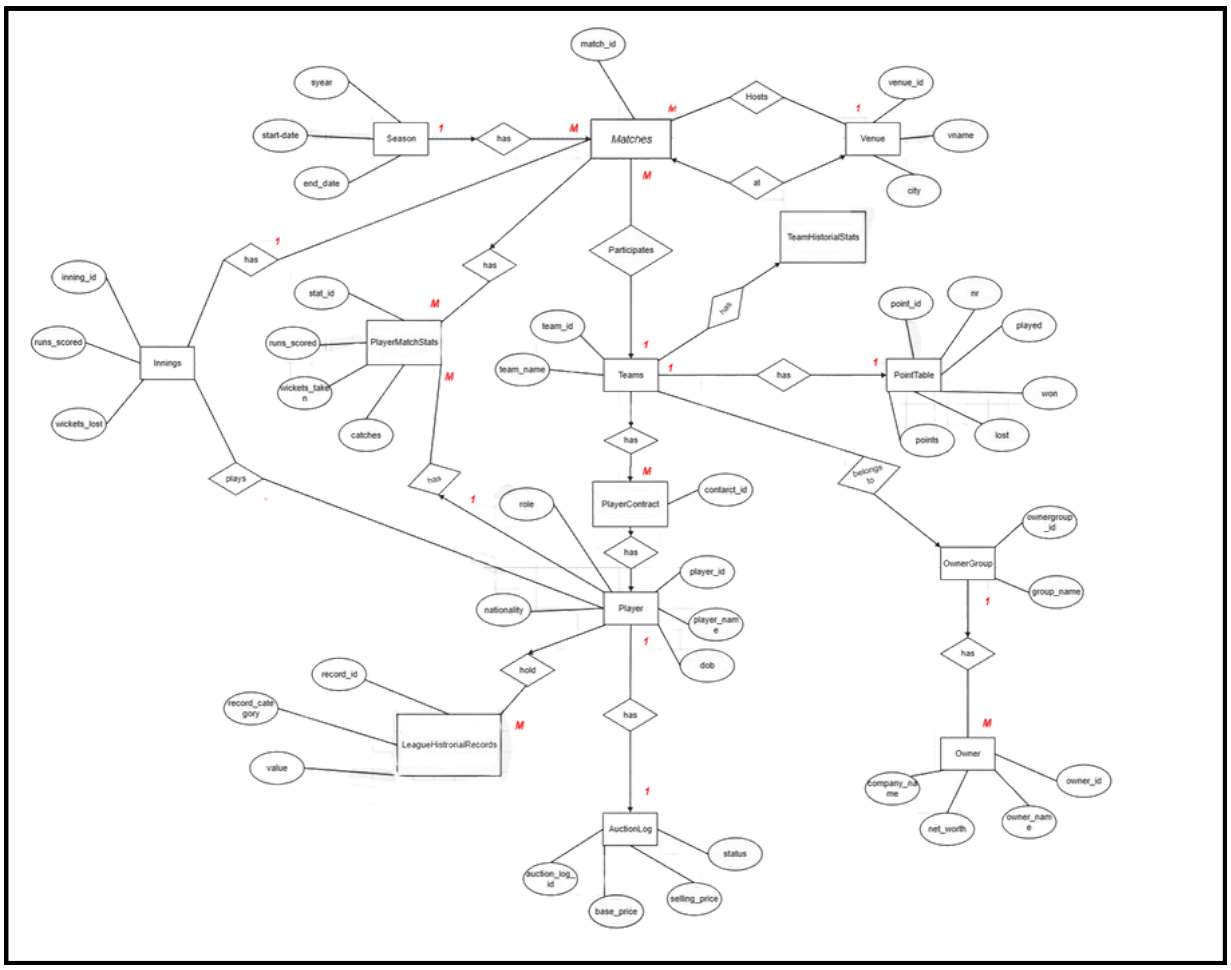


Data Model

Entities

1. **Seasons** (season_year (PK), start_date, end_date)
2. **Teams** (team_id (PK), team_name, short_name, owner_group_id (FK))
3. **Players** (player_id (PK), player_name, dob, nationality, playing_role)
4. **Venues** (venue_id (PK), venue_name, city)
5. **Matches** (match_id (PK), season_year (FK), date, venue_id (FK), team1_id (FK), team2_id (FK), winner_id (FK), toss_winner_id (FK), man_of_the_match_id (FK))
6. **PlayerContracts** (contract_id (PK), player_id (FK), team_id (FK), season_year (FK), salary, status ['Sold', 'Retained', 'Unsold_Pool'])
7. **AuctionLog** (auction_log_id (PK), player_id (FK), season_year (FK), base_price, sold_price, selling_team_id (FK), status ['Sold','Unsold','Retained_Priced'])
8. **Innings** (inning_id (PK), match_id (FK), batting_team_id (FK), runs_scored, wickets_lost)
9. **PlayerMatchStats** (stat_id (PK), match_id (FK), player_id (FK), runs_scored, wickets_taken, catches, etc.)
10. **PointsTable** (point_id (PK), season_year (FK), team_id (FK), played, won, lost, nr, points, nrr)
11. **NewsAnnouncements** (news_id (PK), title, content, published_date, category)
12. **TeamHistoricalStats** (team_id (FK), metric_name, metric_value) - To store all-time team stats
13. **LeagueHistoricalRecords** (record_id (PK), record_category, player_id (FK), value, description) - For all-time league leaders
14. **Owners** (owner_id (PK), owner_name, company_name, net_worth)
15. **OwnerGroups** (owner_group_id (PK), group_name)
16. **TeamOwners** (team_id (FK), owner_id (FK)) - Linking table
17. **Admins** (admin_id (PK), username, password_hash)

ER Diagram


match_id
syear
start-date
Season
end_date
has
Matches
Hosts
at
venue_id
Venue
vname
city
Participates
TeamHistorialStats
has
inning_id
runs_scored
Innings
wickets_lost
stat_id
runs_scored
PlayerMatchStats
wickets_taken
catches
team_id
team_name
Teams
has
point_id
nr
played
won
points
lost
PointTable
has
plays
role
PlayerContract
contarct_id
has
belongs to
ownergroup_id
OwnerGroup
group_name
has
player_id
nationality
Player
player_name
dob
hold
record_id
record_category
LeagueHistorialRecords
value
has
Owner
company_name
net_worth
owner_name
owner_id
AuctionLog
status
auction_log_id
base_price
selling_price

FD and Normalization

- **Players:** (player_id (PK), player_name, dob, nationality, playing_role, is_capped, is_overseas)
- **Teams:** (team_id (PK), team_name, short_name, home_venue_id (FK), owner_group_id (FK))
- **Seasons:** (season_year (PK), season_name, start_date, end_date)
- **PlayerContracts:** (contract_id (PK), player_id (FK), team_id (FK), season_year (FK), salary, status ['Sold', 'Retained', 'Unsold_Pool'])
  - Candidate Key: {player_id, season_year}
- **Matches:** (match_id (PK), season_year (FK), match_datetime, venue_id (FK), team1_id (FK), team2_id (FK), winner_id (FK), man_of_the_match_id (FK))
- **Venues:** (venue_id (PK), venue_name, city)
- **Owners:** (owner_id (PK), owner_name, company_name, net_worth_text)
- **OwnerGroups:** (owner_group_id (PK), group_name)
- **TeamOwnership** (Illustrative Linking Table if an OwnerGroup could own multiple teams, or for complex ownership structures, though our Teams table has a 1:N from OwnerGroups): (owner_group_id (FK), team_id (FK))
- **PlayerMatchStats:** (stat_id (PK), match_id (FK), player_id (FK), inning_number, runs_scored, wickets_taken, etc.)
  - Candidate Key: {match_id, player_id, inning_number}

FD and Normalization

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves decomposing tables into smaller, well-structured relations. This minimizes insertion, deletion, and update anomalies.

1. **First Normal Form:**

If a relation contains a multi-valued attribute, then it violates first normal form, or a relation is in first normal form if it does not contain any multi-valued attribute. A relation is in first normal form if every attribute in that relation is a single-valued attribute.

Teams Table

TeamId	TeamName	OwnerName
T01	Kolkata Knight Riders	Shah Rukh Khand
T01	Kolkata Knight Riders	Juhi Chawla
T01	Kolkata Knight Riders	Jay Mehta
T02	Royal Challengers Bangalore	Prathmesh Verma

The table teams has a multivalued attribute owners hence it is not in 1NF

So we reduce the table to 1NF with combination of Team_name and Owners as Key attribute.

OwnerGroups Table

OwnerGroupId	GroupName
OG01	Mehta Group

Owners Table

OwnerID	OwnerName	OwnerGroupId
Own01	Shah Rukh Khan	OG01
Own02	Juhi Chawla	OG01
Own03	Jay Mehta	OG01

All our defined tables (Players, Teams, Matches, etc.) ensure each cell holds a single value.

Players

PlayerID	PlayerName	PlayerNationality
P001	Virat K.	Indian

PlayerSeasonContracts

PlayerID	SeasonYear	TeamID	Salary
P001	2025	T01	15 crore

Player Phone Numbers

PlayerID	PlayerName	PhoneNumbers
P101	Virat K.	98XXXX0
P101	Virat K.	99XXXX1

Players

PlayerID	PlayerName
P101	Virat K.

PlayerPhoneNumbers

PlayerID	PhoneNumbers
P101	98XXXX0
P101	99XXXX1

Match Fixture With Multiple Umpires in one Column

MatchID	HomeTeam	Awayteam	UmpiresList
M001	CSK	MI	UmpA_Nitin M
M001	CSK	MI	UmpB_Anil C

Matches

MatchID	HomeTeam	AwayTeam
M001	CSK	MI

MatchUmpires

MatchID	UmpireName	UmpireRole
M001	UmpA_NitiM	On-Field
M001	UmpB_AnilC	On-Field

Team With List of Support Staff in One Column

TeamID	TeamName	SupportStaffList
T01	Challenger	Coach_X
T01	Challengers	Physio_Y
T01	Challengers	Analyst_Z

Teams

TeamID	TeamName
T01	Challengers

TeamSupportStaff

TeamID	StaffName	StaffCoach
T01	Coach_X	HeadCoach
T01	Physio_Y	Physiotherapist
T01	Analyst_Z	Data Analyst

Player Awards In Single Season

PlayerID	SeasonYear	AwardsWonInSeason
P101	2023	MoM_Match5
P101	2023	MoM_Match12
P101	2023	MVP

PlayerSeasonAwards

PlayerID	SeasonYear	AwardType	MatchID
P!01	2023	ManOfTheMatch	M005
P101	2023	ManOfTheMatch	M012
P101	2023	MostValuablePl	NULL

2NF

EXAMPLE 1

Match Scores and Player Details

MatchID	PlayerID	PlayerName	RunsScored	WicketsTeam
M01	P101	Virat K.	50	0
M01	P102	Jasprit B.	5	3

Players

PlayerID	PlayerName
P101	Virat K.
P102	Jasprit B.

MatchPlayerStats

MatchID	PlayerID	RunsScored	WicketsTaken
M01	P101	50	0
M01	P102	5	3

EXAMPLE2

Warehouse Part Location

WarehouseID	PartID	WarehouseLocation	PartName	Quantity
W01	P001	Mumbai	Bolt	100
W01	P002	Mumbai	Nut	200

Warehouses

WarehousID	WarehouseLocation
WO1	Mumbai

Parts

PartID	PartName
P001	Bolt
P002	Nut

Inventory

WarehouseID	PartID	Quantity
W01	P001	100
W01	P002	200

EXAMPLE3

Project Assignments and Employment Department

ProjectID	EmployeeID	EmployeeName	EmployeeDept	HoursWorked
PRJ1	E101	Alice	Tech	40
PRJ1	E102	Bob	Sales	30

Employees

EmployeeID	EmployeeName	EmployeeDept
E101	Alice	Tech
E102	Bob	Sales

ProjectHours

ProjectID	EmployeeID	HoursWorked
PRJ1	E101	40
PRJ1	E102	30

3 NF

EXAMPLE1

Player and Team's Home City

PlayerContractID	PlayerID	TeamID	TeamCity	Salary
C001	P101	T01	Bengaluru	15 Cr

PlayerContracts

PlayerContractID	PlayerID	TeamID	Salary
C001	P101	T01	15 Cr

Teams

TeamID	TeamName	TeamCity
T01	RCB	Bengaluru

EXAMPLE 2

Student and Course Department

EnrollmentID	StudentID	CourseID	CourseDeptName	Grade
E01	S101	CS101	CompSci	A

Enrollments

EnrollmentID	StudentID	CourseID	Grade
E01	S101	CS101	A

Courses

CoursesID	CoursesName	CourseDeptName
CS101	Intro CS	CompSci

EXAMPLE 3

Order and Customer's City

OrderID	CustomerID	CustomerCity	OrderTotal
ORD01	CUST1	New York	100.50

Orders

OrderID	CustomerID	OrderTotal
ORD01	CUST1	100.50

Customers

CustomerID	CustomerName	CustomerCity
CUST1	John Doe	New York

BCNF

EXAMPLE 1

Player, Skill and Skil-Specific Coach

PlayerSkillCoach

PlayerID	Skill	CoachName

EXAMPLE2

StudentCourseProfessor

StudentID	CourseID	ProfessorName

EXAMPLE3
Employee, Project, Project Lead

EmpID	ProjectID	ProjectLeadEmpID

4NF

EXAMPLE 1: Player and Their Preferred Playing Positions

PlayerPositionsTeams

PlayerID	PreferredPosition	CurrentTeamID

EXAMPLE 2

EmployeeSkillsLanguages

EmpID	Skill	LanguageSpoken

EXAMPLE 3

CourseDetails

CourseID	RecommendedBook	PrerequisiteCourseID

5NF

EXAMPLE1

AgentCompanyProductionSales

Agent	Company	Product

EXAMPLE 2

PlayerTeamCoachTrial

PlayerID	TeamID	CoachID	SeasonYear

EXAMPLE3

MeetingAttendanceRole

MeetingID	AttendeeID	RoleInMeeting

Table Creation

Admin

```
CREATE TABLE Admins (
    admin_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    username VARCHAR2(100) UNIQUE NOT NULL,
    password_hash VARCHAR2(255) NOT NULL,
    role VARCHAR2(50) DEFAULT 'Editor' CHECK (role IN ('SuperAdmin', 'Editor', 'ScoreKeeper')),
    last_login TIMESTAMP WITH TIME ZONE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

desc Admins;
```

```
Table ADMINS created.

Elapsed: 00:00:00.018
```

Name	Null?	Type
ADMIN_ID	NOT NULL	NUMBER
USERNAME	NOT NULL	VARCHAR2(100)
PASSWORD_HASH	NOT NULL	VARCHAR2(255)
ROLE		VARCHAR2(50)
LAST_LOGIN		TIMESTAMP(6) WITH TIME ZONE
CREATED_AT		TIMESTAMP(6) WITH TIME ZONE

Player

```
CREATE TABLE Players (
    player_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    player_name VARCHAR2(150) NOT NULL UNIQUE,
    dob DATE,
    nationality VARCHAR2(50),
    playing_role VARCHAR2(100),
    is_capped NUMBER(1) CHECK (is_capped IN (0, 1)),
    is_overseas NUMBER(1) DEFAULT 0 CHECK (is_overseas IN (0, 1)),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);
```

```
SQL> desc Players
```

Name	Null?	Type
PLAYER_ID	NOT NULL	NUMBER
PLAYER_NAME	NOT NULL	VARCHAR2(150)
DOB		DATE
NATIONALITY		VARCHAR2(50)
PLAYING_ROLE		VARCHAR2(100)
IS_CAPPED		NUMBER(1)
IS_OVERSEAS		NUMBER(1)
CREATED_AT		TIMESTAMP(6) WITH TIME ZONE
UPDATED_AT		TIMESTAMP(6) WITH TIME ZONE

Player Contract

```
CREATE TABLE PlayerContracts (
    contract_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    player_id NUMBER NOT NULL REFERENCES Players(player_id),
    team_id NUMBER REFERENCES Teams(team_id),
    season_year NUMBER NOT NULL REFERENCES Seasons(season_year),
    salary NUMBER(17,2),
    status VARCHAR2(50) NOT NULL CHECK (status IN ('Sold', 'Retained', 'Unsold_Pool', 'Released', 'Active_Squad')),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT uq_player_season_contract UNIQUE (player_id, season_year)
);
```

```
Table PLAYERCONTRACTS created.

Elapsed: 00:00:00.026
```

```
SQL> desc PlayerContracts


Name        Null?    Type
----------- -------- --------------------------
CONTRACT_ID NOT NULL NUMBER
PLAYER_ID   NOT NULL NUMBER
TEAM_ID              NUMBER
SEASON_YEAR NOT NULL NUMBER
SALARY               NUMBER(17,2)
STATUS      NOT NULL VARCHAR2(50)
CREATED_AT           TIMESTAMP(6) WITH TIME ZONE
```

Seasons

```
CREATE TABLE Seasons (
    season_year NUMBER PRIMARY KEY,
    season_name VARCHAR2(100) NOT NULL,
    start_date DATE,
    end_date DATE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);
```

```
SQL> desc Seasons


Name        Null?    Type
----------- -------- ----------------------------
SEASON_YEAR NOT NULL NUMBER(38)
SEASON_NAME NOT NULL VARCHAR2(100)
START_DATE           DATE
END_DATE             DATE
CREATED_AT           TIMESTAMP(6) WITH TIME ZONE
```

Team

```
CREATE TABLE Teams (
    team_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    team_name VARCHAR2(100) UNIQUE NOT NULL,
    short_name VARCHAR2(10) UNIQUE NOT NULL,
    logo_url VARCHAR2(255),
    home_venue_id NUMBER REFERENCES Venues(venue_id),
    owner_group_id NUMBER REFERENCES OwnerGroups(owner_group_id),
    purchase_year NUMBER,
    funds_remaining_auction2025 NUMBER(17,2),
    overseas_players_squad2025 NUMBER,
    total_players_squad2025 NUMBER,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);
```

```
SQL> desc Teams

Name                          Null?    Type
----------------------------- -------- ----------------------------
TEAM_ID                       NOT NULL NUMBER
TEAM_NAME                     NOT NULL VARCHAR2(100)
SHORT_NAME                    NOT NULL VARCHAR2(10)
LOGO_URL                               VARCHAR2(255)
HOME_VENUE_ID                          NUMBER
OWNER_GROUP_ID                         NUMBER
PURCHASE_YEAR                          NUMBER
FUNDS_REMAINING_AUCTION2025            NUMBER(17,2)
OVERSEAS_PLAYERS_SQUAD2025             NUMBER
TOTAL_PLAYERS_SQUAD2025                NUMBER
CREATED_AT                             TIMESTAMP(6) WITH TIME ZONE
```

Owners

```
CREATE TABLE Owners (
    owner_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    owner_name VARCHAR2(255) NOT NULL,
    owner_group_id NUMBER REFERENCES OwnerGroups(owner_group_id),
    company_name VARCHAR2(255),
    net_worth_text VARCHAR2(255),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);
```

```
SQL> desc Owners_

Name           Null?    Type
-------------- -------- ----------------------------
OWNER_ID       NOT NULL NUMBER
OWNER_NAME     NOT NULL VARCHAR2(255)
OWNER_GROUP_ID          NUMBER
COMPANY_NAME            VARCHAR2(255)
NET_WORTH_TEXT          VARCHAR2(255)
CREATED_AT              TIMESTAMP(6) WITH TIME ZONE
```

OwnersGroups

```
CREATE TABLE OwnerGroups (
    owner_group_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    group_name VARCHAR2(255) NOT NULL UNIQUE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);
```

```
SQL> desc OwnerGroups

Name           Null?    Type
-------------- -------- ----------------------------
OWNER_GROUP_ID NOT NULL NUMBER
GROUP_NAME     NOT NULL VARCHAR2(255)
CREATED_AT              TIMESTAMP(6) WITH TIME ZONE
```

Venue

```
CREATE TABLE Venues (
    venue_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    venue_name VARCHAR2(255) NOT NULL UNIQUE,
    city VARCHAR2(100),
    country VARCHAR2(100) DEFAULT 'India',
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);
```

```
SQL> desc Venues


Name       Null?    Type
---------- -------- --------------------------
VENUE_ID   NOT NULL NUMBER
VENUE_NAME NOT NULL VARCHAR2(255)
CITY                VARCHAR2(100)
COUNTRY             VARCHAR2(100)
CREATED_AT          TIMESTAMP(6) WITH TIME ZONE
```

AuctionLog

```
CREATE TABLE AuctionLog (
    auction_log_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    player_id NUMBER NOT NULL REFERENCES Players(player_id),
    season_year NUMBER NOT NULL REFERENCES Seasons(season_year),
    base_price NUMBER(17,2),
    sold_price NUMBER(17,2) NULL,
    retention_price NUMBER(17,2) NULL,
    winning_team_id NUMBER REFERENCES Teams(team_id) NULL,
    status VARCHAR2(20) NOT NULL CHECK (status IN ('Sold', 'Unsold', 'Retained')),
    rtm_used NUMBER(1) DEFAULT 0 CHECK (rtm_used IN (0,1)),
    auction_timestamp TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT uq_player_season_auction UNIQUE(player_id, season_year)
);
```

```
SQL> desc AuctionLog


Name                Null?    Type
------------------- -------- --------------------------
AUCTION_LOG_ID      NOT NULL NUMBER
PLAYER_ID           NOT NULL NUMBER
SEASON_YEAR         NOT NULL NUMBER
BASE_PRICE                   NUMBER(17,2)
SOLD_PRICE                   NUMBER(17,2)
RETENTION_PRICE              NUMBER(17,2)
WINNING_TEAM_ID              NUMBER
STATUS              NOT NULL VARCHAR2(20)
RTM_USED                     NUMBER(1)
AUCTION_TIMESTAMP            TIMESTAMP(6) WITH TIME ZONE
```

Matches

```
CREATE TABLE Matches (
    match_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    season_year NUMBER NOT NULL REFERENCES Seasons(season_year),
    match_number_league NUMBER,
    match_type VARCHAR2(50) DEFAULT 'League' CHECK (match_type IN ('League', 'Qualifier 1', 'Eliminator', 'Qualifier 2', 'Final')),
    match_datetime TIMESTAMP WITH TIME ZONE NOT NULL,
    venue_id NUMBER REFERENCES Venues(venue_id),
    team1_id NUMBER NOT NULL REFERENCES Teams(team_id),
    team2_id NUMBER NOT NULL REFERENCES Teams(team_id),
    toss_winner_id NUMBER REFERENCES Teams(team_id),
    toss_decision VARCHAR2(10) CHECK (toss_decision IN ('Bat', 'Field')),
    winner_id NUMBER REFERENCES Teams(team_id),
    result_description VARCHAR2(255),
    man_of_the_match_id NUMBER REFERENCES Players(player_id),
    match_status VARCHAR2(20) DEFAULT 'Scheduled' CHECK (match_status IN ('Scheduled', 'Live', 'Completed', 'Abandoned', 'Postponed')),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT chk_teams_different CHECK (team1_id <> team2_id)
);
```

```
desc Matches;
```

```
Name                 Null?    Type
-------------------- -------- ---------------------------
MATCH_ID             NOT NULL NUMBER
SEASON_YEAR          NOT NULL NUMBER
MATCH_NUMBER_LEAGUE           NUMBER
MATCH_TYPE                    VARCHAR2(50)
MATCH_DATETIME       NOT NULL TIMESTAMP(6) WITH TIME ZONE
VENUE_ID                      NUMBER
TEAM1_ID             NOT NULL NUMBER
TEAM2_ID             NOT NULL NUMBER
TOSS_WINNER_ID                NUMBER
TOSS_DECISION                 VARCHAR2(10)
WINNER_ID                     NUMBER
RESULT_DESCRIPTION            VARCHAR2(255)
MAN_OF_THE_MATCH_ID           NUMBER
MATCH_STATUS                  VARCHAR2(20)
CREATED_AT                    TIMESTAMP(6) WITH TIME ZONE
UPDATED_AT                    TIMESTAMP(6) WITH TIME ZONE
```

Innings

```
CREATE TABLE Innings (
    inning_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    match_id NUMBER NOT NULL REFERENCES Matches(match_id) ON DELETE CASCADE,
    inning_number NUMBER NOT NULL CHECK (inning_number IN (1, 2, 3, 4)),
    batting_team_id NUMBER NOT NULL REFERENCES Teams(team_id),
    bowling_team_id NUMBER NOT NULL REFERENCES Teams(team_id),
    runs_scored NUMBER DEFAULT 0,
    wickets_lost NUMBER DEFAULT 0,
    overs_bowled NUMBER(4,1) DEFAULT 0.0,
    extras_awarded NUMBER DEFAULT 0,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT uq_match_inning UNIQUE (match_id, inning_number),
    CONSTRAINT chk_bat_bowl_teams_diff CHECK (batting_team_id <> bowling_team_id)
);
```

```
SQL> desc Innings


Name            Null?    Type
--------------- -------- ---------------------------
INNING_ID       NOT NULL NUMBER
MATCH_ID        NOT NULL NUMBER
INNING_NUMBER   NOT NULL NUMBER
BATTING_TEAM_ID NOT NULL NUMBER
BOWLING_TEAM_ID NOT NULL NUMBER
RUNS_SCORED              NUMBER
WICKETS_LOST             NUMBER
OVERS_BOWLED             NUMBER(4,1)
EXTRAS_AWARDED           NUMBER
CREATED_AT               TIMESTAMP(6) WITH TIME ZONE
```

PlayerMatchStats

```
CREATE TABLE PlayerMatchStats (
    stat_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    match_id NUMBER NOT NULL REFERENCES Matches(match_id) ON DELETE CASCADE,
    player_id NUMBER NOT NULL REFERENCES Players(player_id),
    team_id NUMBER NOT NULL REFERENCES Teams(team_id),
    inning_number NUMBER NOT NULL CHECK (inning_number IN (1,2,3,4)),
    -- Batting
    runs_scored NUMBER, balls_faced NUMBER, fours NUMBER, sixes NUMBER, not_out NUMBER(1) CHECK (not_out IN (0,1)),
    -- Bowling
    overs_bowled NUMBER(3,1), maidens NUMBER, runs_conceded NUMBER, wickets_taken NUMBER,
    bowling_bbi_runs NUMBER, bowling_bbi_wickets NUMBER,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT uq_player_match_inning_stats UNIQUE (match_id, player_id, inning_number)
);
```

```
desc PlayerMatchStats;
```

```
Name                 Null?    Type
-------------------- -------- ----------------------------
STAT_ID              NOT NULL NUMBER
MATCH_ID             NOT NULL NUMBER
PLAYER_ID            NOT NULL NUMBER
TEAM_ID              NOT NULL NUMBER
INNING_NUMBER        NOT NULL NUMBER
RUNS_SCORED                   NUMBER
BALLS_FACED                   NUMBER
FOURS                         NUMBER
SIXES                         NUMBER
NOT_OUT                       NUMBER(1)
OVERS_BOWLED                  NUMBER(3,1)
MAIDENS                       NUMBER
RUNS_CONCEDED                 NUMBER
WICKETS_TAKEN                 NUMBER
BOWLING_BBI_RUNS              NUMBER
BOWLING_BBI_WICKETS           NUMBER
CREATED_AT                    TIMESTAMP(6) WITH TIME ZONE
```

PointsTable

```
CREATE TABLE PointsTable (
    points_table_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    season_year NUMBER NOT NULL REFERENCES Seasons(season_year),
    team_id NUMBER NOT NULL REFERENCES Teams(team_id),
    matches_played NUMBER DEFAULT 0,
    wins NUMBER DEFAULT 0,
    losses NUMBER DEFAULT 0,
    ties NUMBER DEFAULT 0,
    no_result NUMBER DEFAULT 0,
    points NUMBER DEFAULT 0,
    nrr NUMBER(9,3) DEFAULT 0.000,
    runs_for NUMBER DEFAULT 0,
    overs_for NUMBER(6,1) DEFAULT 0.0,
    runs_against NUMBER DEFAULT 0,
    overs_against NUMBER(6,1) DEFAULT 0.0,
    recent_form VARCHAR2(20),
    last_updated TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT uq_season_team_points UNIQUE (season_year, team_id)
);
```

```
desc PointsTable;
```

```
Name            Null?    Type
--------------- -------- ----------------------------
POINTS_TABLE_ID NOT NULL NUMBER
SEASON_YEAR     NOT NULL NUMBER
TEAM_ID         NOT NULL NUMBER
MATCHES_PLAYED           NUMBER
WINS                     NUMBER
LOSSES                   NUMBER
TIES                     NUMBER
NO_RESULT                NUMBER
POINTS                   NUMBER
NRR                      NUMBER(9,3)
RUNS_FOR                 NUMBER
OVERS_FOR                NUMBER(6,1)
RUNS_AGAINST             NUMBER
OVERS_AGAINST            NUMBER(6,1)
RECENT_FORM              VARCHAR2(20)
LAST_UPDATED             TIMESTAMP(6) WITH TIME ZONE
```

NewsAnnouncements

```
CREATE TABLE NewsAnnouncements (
    news_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    title VARCHAR2(500) NOT NULL,
    content CLOB,
    published_datetime TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    category VARCHAR2(100),
    related_match_id NUMBER REFERENCES Matches(match_id) NULL,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);
```

```
SQL> desc NewsAnnouncements


Name                 Null?    Type
-------------------- -------- ---------------------------
NEWS_ID              NOT NULL NUMBER
TITLE                NOT NULL VARCHAR2(500)
CONTENT                       CLOB
PUBLISHED_DATETIME            TIMESTAMP(6) WITH TIME ZONE
CATEGORY                      VARCHAR2(100)
RELATED_MATCH_ID              NUMBER
CREATED_AT                    TIMESTAMP(6) WITH TIME ZONE
```

TeamHistoricalStats

```
CREATE TABLE TeamHistoricalStats (
    team_id NUMBER PRIMARY KEY REFERENCES Teams(team_id),
    all_time_played NUMBER, all_time_won NUMBER, all_time_lost NUMBER, all_time_tied NUMBER, all_time_no_result NUMBER,
    highest_total_runs NUMBER, highest_total_wickets_lost NUMBER,
    lowest_total_runs NUMBER, lowest_total_wickets_lost NUMBER,
    avg_runs_scored_per_match NUMBER(7,2),
    avg_wickets_lost_per_match NUMBER(5,2),
    most_runs_by_player_value NUMBER,
    most_wickets_by_player_value NUMBER,
    highest_individual_score_value NUMBER,
    best_bowling_wickets NUMBER, best_bowling_runs_conceded NUMBER,
    championship_wins VARCHAR2(100), -- Store as comma-separated string
    last_updated TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);
```

```
desc TeamHistoricalStats;
```

```
Name                          Null?    Type
----------------------------- -------- --------------------------
TEAM_ID                       NOT NULL NUMBER
ALL_TIME_PLAYED                        NUMBER
ALL_TIME_WON                           NUMBER
ALL_TIME_LOST                          NUMBER
ALL_TIME_TIED                          NUMBER
ALL_TIME_NO_RESULT                     NUMBER
HIGHEST_TOTAL_RUNS                     NUMBER
HIGHEST_TOTAL_WICKETS_LOST             NUMBER
LOWEST_TOTAL_RUNS                      NUMBER
LOWEST_TOTAL_WICKETS_LOST              NUMBER
AVG_RUNS_SCORED_PER_MATCH              NUMBER(7,2)
AVG_WICKETS_LOST_PER_MATCH             NUMBER(5,2)
MOST_RUNS_BY_PLAYER_VALUE              NUMBER
MOST_WICKETS_BY_PLAYER_VALUE           NUMBER
HIGHEST_INDIVIDUAL_SCORE_VALUE         NUMBER
BEST_BOWLING_WICKETS                   NUMBER
BEST_BOWLING_RUNS_CONCEDED             NUMBER
```

LeagueHistoricalRecords

```
CREATE TABLE LeagueHistoricalRecords (
    record_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    category VARCHAR2(100) NOT NULL UNIQUE,
    player_id NUMBER REFERENCES Players(player_id) NULL,
    record_value_numeric NUMBER(12,2) NULL,
    record_value_text VARCHAR2(255) NULL,
    description CLOB,
    last_updated TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);
```

```
SQL> desc LeagueHistoricalRecords


Name                 Null?    Type
-------------------- -------- --------------------------
RECORD_ID            NOT NULL NUMBER
CATEGORY             NOT NULL VARCHAR2(100)
PLAYER_ID                     NUMBER
RECORD_VALUE_NUMERIC          NUMBER(12,2)
RECORD_VALUE_TEXT             VARCHAR2(255)
DESCRIPTION                   CLOB
LAST_UPDATED                  TIMESTAMP(6) WITH TIME ZONE
```

AuditLog

```
CREATE TABLE AuditLog (
    log_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    table_name VARCHAR2(100) NOT NULL,
    record_pk_value VARCHAR2(100),
    action_type VARCHAR2(10) NOT NULL CHECK (action_type IN ('INSERT', 'UPDATE', 'DELETE')),
    admin_id NUMBER REFERENCES Admins(admin_id) NULL,
    change_timestamp TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    old_data CLOB,
    new_data CLOB
);

desc AuditLog;
```

```
Name             Null?    Type
---------------- -------- --------------------------
LOG_ID           NOT NULL NUMBER
TABLE_NAME       NOT NULL VARCHAR2(100)
RECORD_PK_VALUE           VARCHAR2(100)
ACTION_TYPE      NOT NULL VARCHAR2(10)
ADMIN_ID                  NUMBER
CHANGE_TIMESTAMP          TIMESTAMP(6) WITH TIME ZONE
OLD_DATA                  CLOB
NEW_DATA                  CLOB
```

Insertion of Values

```
INSERT INTO Seasons (season_year, season_name, start_date, end_date)
VALUES (2025, 'TATA IPL 2025', TO_DATE('2025-03-22', 'YYYY-MM-DD'), TO_DATE('2025-05-25', 'YYYY-MM-DD'));
```

```
SQL> INSERT INTO Seasons (season_year, season_name, start_date, end_date)
     VALUES (2025, 'TATA IPL 2025', TO_DATE('2025-03-22', 'YYYY-MM-DD'), TO_DATE('2025-05-25', 'YYYY-MM-DD'))



1 row inserted.
```

```
INSERT INTO Venues (venue_name, city) VALUES ('M. A. Chidambaram Stadium', 'Chennai');
INSERT INTO Venues (venue_name, city) VALUES ('Eden Gardens', 'Kolkata');
INSERT INTO Venues (venue_name, city) VALUES ('Narendra Modi Stadium', 'Ahmedabad');
INSERT INTO Venues (venue_name, city) VALUES ('Arun Jaitley Stadium', 'Delhi');
INSERT INTO Venues (venue_name, city) VALUES ('BRSABV Ekana Cricket Stadium', 'Lucknow');
INSERT INTO Venues (venue_name, city) VALUES ('Wankhede Stadium', 'Mumbai');
INSERT INTO Venues (venue_name, city) VALUES ('PCA New Stadium, Mullanpur', 'Mullanpur');
INSERT INTO Venues (venue_name, city) VALUES ('Sawai Mansingh Stadium', 'Jaipur');
INSERT INTO Venues (venue_name, city) VALUES ('M. Chinnaswamy Stadium', 'Bengaluru');
INSERT INTO Venues (venue_name, city) VALUES ('Rajiv Gandhi Intl. Cricket Stadium', 'Hyderabad');
```

```
INSERT INTO OwnerGroups (group_name) VALUES ('Reliance Industries');
INSERT INTO OwnerGroups (group_name) VALUES ('Mehta Group');
INSERT INTO Owners (owner_name, owner_group_id, company_name, net_worth_text) VALUES
('Mukesh Ambani', (SELECT owner_group_id FROM OwnerGroups WHERE group_name = 'Reliance Industries'), 'Reliance Industries', '$92.8 billion');
INSERT INTO Owners (owner_name, owner_group_id, company_name, net_worth_text) VALUES
('Shah Rukh Khan', (SELECT owner_group_id FROM OwnerGroups WHERE group_name = 'Mehta Group'), 'Mehta Group', '$780 million');
INSERT INTO Owners (owner_name, owner_group_id, company_name, net_worth_text) VALUES
('Juhi Chawla', (SELECT owner_group_id FROM OwnerGroups WHERE group_name = 'Mehta Group'), 'Mehta Group', '$6 million');
```

```
INSERT INTO OwnerGroups (group_name) VALUES ('India Cements');
```

```
INSERT INTO Teams (team_name, short_name, home_venue_id, owner_group_id, purchase_year,
 funds_remaining_auction2025, overseas_players_squad2025, total_players_squad2025) VALUES
('Mumbai Indians', 'MI', (SELECT venue_id FROM Venues
WHERE venue_name = 'Wankhede Stadium'),
(SELECT owner_group_id FROM OwnerGroups
WHERE group_name = 'Reliance Industries'), 2008, 2000000, 8, 23);
INSERT INTO Teams (team_name, short_name, home_venue_id,
owner_group_id, purchase_year, funds_remaining_auction2025,
 overseas_players_squad2025, total_players_squad2025)
VALUES('Kolkata Knight Riders', 'KKR', (SELECT venue_id FROM Venues WHERE venue_name = 'Eden Gardens'),
(SELECT owner_group_id FROM OwnerGroups
WHERE group_name = 'Mehta Group'), 2008, 500000, 8, 21);
INSERT INTO Teams (team_name, short_name, home_venue_id, owner_group_id, purchase_year,
funds_remaining_auction2025, overseas_players_squad2025, total_players_squad2025) VALUES
('Chennai Super Kings', 'CSK', (SELECT venue_id FROM Venues WHERE venue_name = 'M. A. Chidambaram Stadi
(SELECT owner_group_id FROM OwnerGroups WHERE group_name = 'India Cements'), 2008, 500000, 7, 25);
```

```sql
INSERT INTO Players (player_name, nationality, playing_role, is_capped, is_overseas)
VALUES ('Noor Ahmad', 'Afghanistan', 'Bowler', 1, 1);
INSERT INTO Players (player_name, nationality, playing_role, is_capped, is_overseas)
VALUES ('Ravichandran Ashwin', 'India', 'Bowler', 1, 0);
INSERT INTO Players (player_name, nationality, playing_role, is_capped, is_overseas)
VALUES ('Devon Conway', 'New Zealand', 'Wicketkeeper-Batsman', 1, 1);
INSERT INTO Players (player_name, nationality, playing_role, is_capped, is_overseas)
VALUES ('Ruturaj Gaikwad', 'India', 'Batsman', 1, 0);
INSERT INTO Players (player_name, nationality, playing_role, is_capped, is_overseas)
VALUES ('MS Dhoni', 'India', 'Wicketkeeper-Batsman', 0, 0); -- Uncapped=0
INSERT INTO Players (player_name, nationality, playing_role, is_capped, is_overseas)
VALUES ('David Warner', 'Australia', 'Batsman', 1, 1);
INSERT INTO Players (player_name, nationality, playing_role, is_capped, is_overseas)
VALUES ('Rishabh Pant', 'India', 'Wicketkeeper-Batsman', 1, 0);
INSERT INTO Players (player_name, nationality, playing_role, is_capped, is_overseas)
VALUES ('Surya Kumar Yadav', 'India', 'Batsman', 1, 0);
INSERT INTO Players (player_name, nationality, playing_role, is_capped, is_overseas)
VALUES ('Prasidh Krishna', 'India', 'Bowler', 1, 0);
INSERT INTO Players (player_name, nationality, playing_role, is_capped, is_overseas)
VALUES ('Virat Kohli', 'India', 'Batsman', 1, 0);
INSERT INTO Players (player_name, nationality, playing_role, is_capped, is_overseas)
VALUES ('Yuzvendra Chahal', 'India', 'Bowler', 1, 0);
INSERT INTO Players (player_name, nationality, playing_role, is_capped, is_overseas)
VALUES ('Alzarri Joseph', 'West Indies', 'Bowler', 1, 1);
INSERT INTO Players (player_name, nationality, playing_role, is_capped, is_overseas)
VALUES ('Lungisani Ngidi', 'South Africa', 'Bowler', 1, 1);
INSERT INTO Players (player_name, nationality, playing_role, is_capped, is_overseas)
VALUES ('Tim David', 'Australia', 'All-Rounder', 1, 1);
```

```sql
INSERT INTO AuctionLog (player_id, season_year, base_price, sold_price,
 winning_team_id, status, rtm_used) VALUES
((SELECT player_id FROM Players
WHERE player_name = 'Noor Ahmad'), 2025, 20000000, 100000000,
 (SELECT team_id FROM Teams WHERE short_name = 'CSK'), 'Sold', 0);

INSERT INTO AuctionLog (player_id, season_year, retention_price,
 winning_team_id, status, rtm_used) VALUES
((SELECT player_id FROM Players
WHERE player_name = 'Ruturaj Gaikwad'), 2025, 180000000,
 (SELECT team_id FROM Teams WHERE short_name = 'CSK'), 'Retained', 0);

INSERT INTO AuctionLog (player_id, season_year, base_price,
 status, rtm_used) VALUES
((SELECT player_id FROM Players
WHERE player_name = 'David Warner'), 2025, 200000000, 'Unsold', 0);


INSERT INTO PlayerContracts (player_id, team_id, season_year,
 salary, status) VALUES
((SELECT player_id FROM Players WHERE player_name = 'Noor Ahmad'),
 (SELECT team_id FROM Teams WHERE short_name = 'CSK'), 2025, 100000000, 'Sold');
INSERT INTO PlayerContracts (player_id, team_id, season_year, salary, status) VALUES
((SELECT player_id FROM Players WHERE player_name = 'Ruturaj Gaikwad'),
 (SELECT team_id FROM Teams WHERE short_name = 'CSK'), 2025, 180000000, 'Retained');
```

```sql
INSERT INTO Matches (season_year, match_number_league, match_type,
 match_datetime, venue_id, team1_id, team2_id, match_status) VALUES
(2025, 1, 'League', TO_TIMESTAMP_TZ('2025-03-22 19:30:00 +05:30', 'YYYY-MM-DD HH24:MI:SS TZH:TZM'),
 (SELECT venue_id FROM Venues WHERE venue_name = 'Eden Gardens'),
  (SELECT team_id FROM Teams WHERE short_name = 'KKR'),
   (SELECT team_id FROM Teams WHERE short_name = 'RCB'), 'Scheduled');

INSERT INTO Matches (season_year, match_number_league, match_type,
 match_datetime, venue_id, team1_id, team2_id, match_status) VALUES
(2025, 2, 'League', TO_TIMESTAMP_TZ('2025-03-23 15:30:00 +05:30', 'YYYY-MM-DD HH24:MI:SS TZH:TZM'),
 (SELECT venue_id FROM Venues WHERE city = 'Hyderabad'),
  (SELECT team_id FROM Teams WHERE short_name = 'SRH'),
   (SELECT team_id FROM Teams WHERE short_name = 'RR'), 'Scheduled');

INSERT INTO Matches (season_year, match_number_league, match_type,
 match_datetime, venue_id, team1_id, team2_id, match_status,
  winner_id, man_of_the_match_id, result_description) VALUES
(2025, 56, 'League', TO_TIMESTAMP_TZ('2025-05-06 19:30:00 +05:30', 'YYYY-MM-DD HH24:MI:SS TZH:TZM'),
 (SELECT venue_id FROM Venues WHERE city = 'Mumbai'),
  (SELECT team_id FROM Teams WHERE short_name = 'MI'),
   (SELECT team_id FROM Teams WHERE short_name = 'GT'), 'Completed',
 (SELECT team_id FROM Teams WHERE short_name = 'GT'),
 (SELECT player_id FROM Players WHERE player_name = 'Surya Kumar Yadav'), 'GT won by X runs');
```

```sql
INSERT INTO PointsTable (season_year, team_id, matches_played,
 wins, losses, no_result, points, nrr, runs_for, overs_for, runs_against, overs_against, recent_form)
  VALUES
(2025,
(SELECT team_id FROM Teams WHERE short_name = 'GT')
, 11, 8, 3, 0, 16, 0.793, 2130, 212.5, 1975, 214.2, 'WWLWW');
INSERT INTO PointsTable (season_year, team_id, matches_played,
 wins, losses, no_result, points, nrr, runs_for, overs_for, runs_against, overs_against, recent_form)
  VALUES
(2025,
(SELECT team_id FROM Teams WHERE short_name = 'RCB')
, 11, 8, 3, 0, 16, 0.482, 1938, 205.1, 1863, 207.5, 'WWWWL');
```

```sql
INSERT INTO PlayerMatchStats (match_id, player_id,
 team_id, inning_number, runs_scored, balls_faced, not_out) VALUES
( (SELECT match_id FROM Matches
WHERE match_number_league=56 AND season_year=2025),
 (SELECT player_id FROM Players WHERE player_name = 'Surya Kumar Yadav'),
  (SELECT team_id FROM Teams WHERE short_name = 'MI'), 1, 68, 40, 1);
INSERT INTO PlayerMatchStats (match_id, player_id,
 team_id, inning_number, wickets_taken, overs_bowled, runs_conceded) VALUES
( (SELECT match_id FROM Matches
WHERE match_number_league=56 AND season_year=2025),
 (SELECT player_id FROM Players WHERE player_name = 'Prasidh Krishna'),
 (SELECT team_id FROM Teams WHERE short_name = 'GT'), 1, 2, 4.0, 30);
```

```
INSERT INTO NewsAnnouncements (title, published_datetime,
 category, related_match_id) VALUES
('Match 56, MI vs GT - Code of Conduct', TO_TIMESTAMP_TZ('2025-05-07 10:00:00 +05:30',
 'YYYY-MM-DD HH24:MI:SS TZH:TZM'), 'Code of Conduct',
  (SELECT match_id FROM Matches WHERE match_number_league=56 AND season_year=2025));
INSERT INTO NewsAnnouncements (title, published_datetime, category) VALUES
('Chennai Super Kings sign Urvil Patel as injury replacement',
TO_TIMESTAMP_TZ('2025-05-05 14:00:00 +05:30', 'YYYY-MM-DD HH24:MI:SS TZH:TZM'), 'Injury Replacement');
```

```
INSERT INTO TeamHistoricalStats (team_id, all_time_played,
 all_time_won, all_time_lost, all_time_tied, all_time_no_result,
  highest_total_runs, highest_total_wickets_lost, lowest_total_runs,
   lowest_total_wickets_lost, avg_runs_scored_per_match, avg_wickets_lost_per_match,
   most_runs_by_player_value, most_wickets_by_player_value, highest_individual_score_value,
   best_bowling_wickets, best_bowling_runs_conceded, championship_wins) VALUES
((SELECT team_id FROM Teams WHERE short_name = 'CSK'),
 251, 140, 108, 0, 3, 246, 5, 71, 3, 163, 5, 4832, 140, 127, 5, 16, '2010,2011,2018,2021,2023');

INSERT INTO TeamHistoricalStats (team_id, all_time_played,
all_time_won, all_time_lost, all_time_tied, all_time_no_result,
 highest_total_runs, highest_total_wickets_lost, lowest_total_runs,
 lowest_total_wickets_lost, avg_runs_scored_per_match, avg_wickets_lost_per_match,
  most_runs_by_player_value, most_wickets_by_player_value, highest_individual_score_value,
   best_bowling_wickets, best_bowling_runs_conceded, championship_wins) VALUES
((SELECT team_id FROM Teams WHERE short_name = 'MI')
, 274, 151, 122, 0, 1, 247, 9, 68, 2, 162, 5, 5758, 178, 114, 6, 12, '2013,2015,2017,2019,2020');
```

```
INSERT INTO LeagueHistoricalRecords (category, player_id, record_value_numeric, description)
 VALUES
('Most Runs All Time',
(SELECT player_id FROM Players
 WHERE player_name = 'Virat Kohli'), 8509, 'Most career runs in IPL history');
INSERT INTO LeagueHistoricalRecords (category, player_id, record_value_text, description)
VALUES
('Best Bowling Figures All Time', (SELECT player_id FROM Players
WHERE player_name = 'Alzarri Joseph'), '6/12', 'Best bowling figures in an IPL innings');
```

```
INSERT INTO Admins (username, password_hash, role)
VALUES ('ARYAK', 'hashed_password_example1', 'SuperAdmin');
INSERT INTO Admins (username, password_hash, role)
 VALUES ('KHUSHI', 'hashed_password_example2', 'Editor');
```

SQL Queries

```
SELECT
    m.match_datetime,
    ht.team_name AS home_team,
    at.team_name AS away_team,
    v.venue_name,
    v.city
FROM Matches m
JOIN Teams ht ON m.team1_id = ht.team_id
JOIN Teams at ON m.team2_id = at.team_id
JOIN Venues v ON m.venue_id = v.venue_id
WHERE m.season_year = 2025
  AND m.match_datetime >=
   TO_TIMESTAMP_TZ('2025-03-24 00:00:00 +05:30', 'YYYY-MM-DD HH24:MI:SS TZH:TZM')
  AND m.match_datetime <
  TO_TIMESTAMP_TZ('2025-04-01 00:00:00 +05:30', 'YYYY-MM-DD HH24:MI:SS TZH:TZM')
ORDER BY m.match_datetime;
```

	MATCH_DATETIME	HOME_TEAM	AWAY_TEAM	VENUE_NAME	CITY
1	2025-03-24T14:00:(	Delhi Capitals	Lucknow Super Giant	Visakhapatnam Stadi	Visakhapatnam
2	2025-03-25T14:00:(	Gujarat Titans	Punjab Kings	Narendra Modi Stadi	Ahmedabad
3	2025-03-27T14:00:(	Sunrisers Hyderabad	Lucknow Super Giant	Rajiv Gandhi Intl. Cric	Hyderabad
4	2025-03-28T14:00:(	Chennai Super Kings	Royal Challengers Be	M. A. Chidambaram S	Chennai
5	2025-03-29T14:00:(	Gujarat Titans	Mumbai Indians	Narendra Modi Stadi	Ahmedabad
6	2025-03-30T10:00:(	Delhi Capitals	Sunrisers Hyderabad	Visakhapatnam Stadi	Visakhapatnam

```
SELECT
    p.player_name,
    al.base_price,
    al.sold_price,
    t.team_name AS bought_by_team
FROM AuctionLog al
JOIN Players p ON al.player_id = p.player_id
JOIN Teams t ON al.winning_team_id = t.team_id
WHERE al.season_year = 2025
  AND al.status = 'Sold'
  AND al.sold_price IS NOT NULL
  AND al.base_price IS NOT NULL
  AND (al.sold_price - al.base_price) >= 50000000
ORDER BY (al.sold_price - al.base_price) DESC;
```

	PLAYER_NAME	BASE_PRICE	SOLD_PRICE	BOUGHT_BY_TEAM
1	Shreyas Iyer	20000000	267500000	Punjab Kings
2	Pat Cummins	20000000	180000000	Sunrisers Hyderabad
3	Yuzvendra Chahal	20000000	160000000	Punjab Kings
4	Hardik Pandya	20000000	160000000	Gujarat Titans
5	Virat Kohli	20000000	150000000	Royal Challengers Be
6	Mitchell Starc	20000000	117500000	Delhi Capitals
7	Noor Ahmad	20000000	100000000	Chennai Super Kings

```sql
WITH RankedPlayerBuys AS (
    SELECT
        t.team_name,
        p.player_name,
        al.sold_price,
        ROW_NUMBER() OVER(PARTITION BY t.team_id
        ORDER BY al.sold_price DESC NULLS LAST) as rn
    FROM AuctionLog al
    JOIN Players p ON al.player_id = p.player_id
    JOIN Teams t ON al.winning_team_id = t.team_id
    WHERE al.season_year = 2025
      AND al.status = 'Sold'
      AND al.sold_price IS NOT NULL
)
SELECT
    team_name,
    player_name AS most_expensive_buy,
    sold_price
FROM RankedPlayerBuys
WHERE rn = 1
ORDER BY sold_price DESC;
```

	TEAM_NAME	MOST_EXPENSIVE_	SOLD_PRICE
1	Punjab Kings	Shreyas Iyer	267500000
2	Sunrisers Hyderabad	Pat Cummins	180000000
3	Gujarat Titans	Hardik Pandya	160000000
4	Royal Challengers Be	Virat Kohli	150000000
5	Delhi Capitals	Mitchell Starc	117500000
6	Chennai Super Kings	Noor Ahmad	100000000

```sql
SELECT
    p.player_name,
    lhr.category AS all_time_record_category,
    al.sold_price AS price_2025
FROM Players p
JOIN LeagueHistoricalRecords lhr ON p.player_id = lhr.player_id
JOIN AuctionLog al ON p.player_id = al.player_id
WHERE al.season_year = 2025 AND al.status = 'Sold';
```

	PLAYER_NAME	ALL_TIME_RECORD_	PRICE_2025
1	Yuzvendra Chahal	Most Wickets All Tim	160000000
2	Virat Kohli	Most Runs All Time	150000000
3	Alzarri Joseph	Best Bowling Figures	30000000

```sql
SELECT
    p.player_name,
    lhr.category AS all_time_record_category
FROM Players p
JOIN LeagueHistoricalRecords lhr ON p.player_id = lhr.player_id
WHERE p.player_id IN (
    SELECT player_id
    FROM AuctionLog
    WHERE season_year = 2025 AND status = 'Sold'
);
```

	PLAYER_NAME	ALL_TIME_RECORD_CATEGORY
1	Yuzvendra Chahal	Most Wickets All Time
2	Virat Kohli	Most Runs All Time
3	Alzarri Joseph	Best Bowling Figures All Time

PL/SQL

PL/SQL Function

The PL/SQL stored function or simply a function is a PL/SQL return value operation which performs one task. It is just like function in other programming languages.

The procedure contains a header with return value.

PL/SQL Procedures

The PL/SQL stored procedure or simply a procedure is a PL/SQL block which performs one or more specific tasks. It is just like procedures in other programming languages.

The procedure contains a header and a body.

- Header: The header contains the name of the procedure and the parameters or variables passed to the procedure.
- Body: The body contains a declaration section, execution section and exception section similar to a general PL/SQL block.

Head-to-Head Comparison Function

```
CREATE OR REPLACE TYPE T_H2H_Result AS OBJECT (
    match_description VARCHAR2(500),
    team1_wins        NUMBER,
    team2_wins        NUMBER,
    ties              NUMBER
);
/

CREATE OR REPLACE TYPE T_H2H_Result_Table AS TABLE OF T_H2H_Result;
/
```

```
CREATE OR REPLACE FUNCTION fn_get_head_to_head_all_time(
    p_team1_name IN VARCHAR2,
    p_team2_name IN VARCHAR2
)
RETURN T_H2H_Result_Table PIPELINED
IS
    v_team1_id NUMBER;
    v_team2_id NUMBER;
    v_team1_all_time_wins NUMBER := 0;
    v_team2_all_time_wins NUMBER := 0;
    v_all_time_ties NUMBER := 0;
    v_winner_id NUMBER;
    v_result_description VARCHAR2(255);
    v_match_description VARCHAR2(500);

    CURSOR match_cursor (c_t1_id NUMBER, c_t2_id NUMBER) IS
        SELECT winner_id, result_description
        FROM Matches
        WHERE ((team1_id = c_t1_id AND team2_id = c_t2_id)
           OR (team1_id = c_t2_id AND team2_id = c_t1_id))
          AND match_status = 'Completed';
```

```
BEGIN
    BEGIN
        SELECT team_id INTO v_team1_id FROM Teams WHERE team_name = p_team1_name
        OR short_name = p_team1_name;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE_APPLICATION_ERROR(-20001, 'Team 1 not found: ' || p_team1_name);
    END;

    BEGIN
        SELECT team_id INTO v_team2_id FROM Teams WHERE team_name = p_team2_name
        OR short_name = p_team2_name;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE_APPLICATION_ERROR(-20002, 'Team 2 not found: ' || p_team2_name);
    END;

    IF v_team1_id = v_team2_id THEN
        RAISE_APPLICATION_ERROR(-20003, 'Cannot compare a team against itself.');
    END IF;
```

```
    OPEN match_cursor(v_team1_id, v_team2_id);
    LOOP
        FETCH match_cursor INTO v_winner_id, v_result_description;
        EXIT WHEN match_cursor%NOTFOUND;

        IF v_winner_id = v_team1_id THEN
            v_team1_all_time_wins := v_team1_all_time_wins + 1;
        ELSIF v_winner_id = v_team2_id THEN
            v_team2_all_time_wins := v_team2_all_time_wins + 1;
        ELSIF v_winner_id IS NULL AND INSTR(UPPER(v_result_description), 'TIE') > 0 THEN
            v_all_time_ties := v_all_time_ties + 1;
        END IF;
    END LOOP;
    CLOSE match_cursor;
```

```
    v_match_description := 'All-Time Head-to-Head: ' || p_team1_name || ' vs ' || p_team2_name;


    PIPE ROW(T_H2H_Result(v_match_description,
     v_team1_all_time_wins, v_team2_all_time_wins, v_all_time_ties));

    RETURN;
END;
/
SELECT * FROM TABLE(fn_get_head_to_head_all_time('Mumbai Indians', 'Chennai Super Kings'));
```

	MATCH_DESCRIPTION	TEAM1_WINS	TEAM2_WINS	TIES
1	All-Time Head-to-Head: Mumbai Indians vs Chennai Super Kings	1	1	0

Procedure to Print Team details

```
1  CREATE or REPLACE procedure print_team(t_name team.team_name%type)
2  AS
3    r_team team%rowtype;
4  BEGIN
5    SELECT * INTO r_team FROM team WHERE team_name = t_name;
6    dbms_output.put_line('Team is ' ||r_team.team_name);
7  dbms_output.put_line(' Having balance ' ||r_team.balance);
8  dbms_output.put_line(' Number of players are ' ||r_team.player_count);
9  dbms_output.put_line(' captain is ' ||r_team.captain);
10 END;
```

```
1  begin
2  print_team('SRH');
3  end;
```

```
Statement processed.
Team is SRH
 Having balance 9240
 Number of players are 1
 captain is Aiden Markram
```

Procedure to Print Player

```
1  Create or replace procedure print_player(p_name player_details.player_name%type)
2  IS
3    r_player player_details%rowtype;
4  BEGIN
5  SELECT * INTO r_player FROM player_details WHERE player_name = p_name;
6  dbms_output.put_line('Player is' ||r_player.player_name);
7  dbms_output.put_line('no of matches played' ||r_player.Matches);
8  dbms_output.put_line('Total Runs' || r_player.Runs);
9  dbms_output.put_line('Batting Avg' ||r_player.Batting_Avg);
10 dbms_output.put_line('Highest score' ||r_player.Best);
11 dbms_output.put_line('Batting Strike Rate' ||r_player.Batting_SR);
12 dbms_output.put_line('No. of wickets taken' ||r_player.Wickets);
13 dbms_output.put_line('Bowling Strike Rate' ||r_player.Bowling_SR);
14 dbms_output.put_line('Bowling Avg' ||r_player.Bowling_Avg);
15 dbms_output.put_line('Status' || r_player.Status);
16 dbms_output.put_line('Player type' ||r_player.Player_type);
17 END;
```

```
Procedure created.
```

```
1  exec print_player('Virat Kohli');
```

```
Statement processed.
Player isVirat Kohli
no of matches played230
Total Runs6903
Batting Avg36.52
Highest score113
Batting Strike Rate129.61
No. of wickets taken4
Bowling Strike Rate62.75
Bowling Avg92
StatusSold
Player typeBatsman
```

Procedure to Sell Player

```
CREATE or REPLACE procedure sell_player(p_name player_details.player_name%type)
AS
    t player_details.team_name%type;
BEGIN
    update player_details set status='Sold' where player_name=p_name;
if SQL%NOTFOUND then
    dbms_output.put_line('Player not present in auction');
else
    select team_name into t from player_details where player_name=p_name;
    dbms_output.put_line('Player succesfully sold to '|| t);
end if;
END;
```

```
Procedure created.
```

Procedure to add new player to Player_details

```
CREATE or REPLACE procedure add_player(p_name player_details.player_name%type,p_price player_details.current_price%type,
    p_matches int,p_runs int,p_sr player_details.batting_Sr%type,p_avg player_details.batting_avg%type,p_b int,
    w int,p_bsr player_details.bowling_sr%type,p_bavg player_details.bowling_avg%type,p_type player_details.player_type%type)
AS
BEGIN
  insert into player_details(player_name,current_price,matches,runs,batting_sr,batting_avg,best,wickets,bowling_sr,bowling_avg,player_type)
    values(p_name,p_price,p_matches,p_runs,p_sr,p_avg,p_b,w,p_bsr,p_bavg,p_type);
END;
```

```
Procedure created.
```

Procedure to bid for a Player

```
CREATE or REPLACE procedure register_bid(p_name player_details.player_name%type,t_name player_details.team_name%type,p int)
AS
BEGIN
    update player_details set team_name=t_name,current_price=p where player_name=p_name;
if SQL%NOTFOUND then
    dbms_output.put_line('Player not present in the auction');
else
    dbms_output.put_line('Bid Registered Succesfully');
end if;
END;
```

```
Procedure created.
```

PL/SQL Trigger

Trigger is invoked by Oracle engine automatically whenever a specified event occurs.Trigger is stored into database and invoked repeatedly, when specific condition match.

Triggers are stored programs, which are automatically executed or fired when some event occurs.

Triggers are written to be executed in response to any of the following events.

- A database manipulation (DML) statement (DELETE, INSERT, or UPDATE).
- A database definition (DDL) statement (CREATE, ALTER, or DROP).
- A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Triggers could be defined on the table, view, schema, or database with which the event is associated.

Trigger to automatically add players in Players table and assign them Player_id

```
1  CREATE OR REPLACE trigger add_player
2  after insert on player_details
3  for each row
4      declare
5      n player.player_id%type;
6  begin
7      select max(player_id) into n from player;
8  n:=n+1;
9  insert into player values(n,:new.player_name);
10 end;
11
```

Contract Sync Trigger

```
CREATE OR REPLACE TRIGGER auction_log_after_update_trigger
AFTER INSERT OR UPDATE OF status, winning_team_id, sold_price ON AuctionLog
FOR EACH ROW
DECLARE
BEGIN
    IF :NEW.status = 'Sold' AND :NEW.winning_team_id
    IS NOT NULL AND :NEW.sold_price IS NOT NULL THEN
        MERGE INTO PlayerContracts pc
        USING (SELECT :NEW.player_id AS player_id, :NEW.season_year AS season_year FROM dual) src
        ON (pc.player_id = src.player_id AND pc.season_year = src.season_year)
        WHEN MATCHED THEN
            UPDATE SET pc.team_id = :NEW.winning_team_id,
                       pc.salary = :NEW.sold_price,
                       pc.status = 'Sold'
        WHEN NOT MATCHED THEN
            INSERT (pc.player_id, pc.team_id, pc.season_year, pc.salary, pc.status)
            VALUES (:NEW.player_id, :NEW.winning_team_id,
             :NEW.season_year, :NEW.sold_price, 'Sold');
    ELSIF :NEW.status = 'Unsold' AND :OLD.status = 'Sold' THEN
        UPDATE PlayerContracts pc
           SET pc.status = 'Unsold_Pool',
               pc.team_id = NULL,
               pc.salary = NULL
         WHERE pc.player_id = :NEW.player_id
           AND pc.season_year = :NEW.season_year;
    END IF;
END;
/
```

```
SQL> CREATE OR REPLACE TRIGGER auction_log_after_update_trigger
     AFTER INSERT OR UPDATE OF status, winning_team_id, sold_price ON AuctionLog
     FOR EACH ROW
     DECLARE...
Show more...



Trigger AUCTION_LOG_AFTER_UPDATE_TRIGGER compiled

Elapsed: 00:00:00.025
```

Here we use the procedure add_player to add details of a new player to player_details table. The trigger then invokes and automatically adds the player to player table with an ID.

Trigger to automatically increase player_count and deduct balance from team table when a player is sold to a team

```
CREATE OR REPLACE trigger team_edit
after update of status on player_details
for each row
begin
if :new.status='Sold' then
    update team set balance=balance-:new.current_price where team_name=:new.team_name;
    update team set player_count=player_count+1 where team_name=:new.team_name;
end if;
end;
```

Here we use the procedure sell_player to change the status of a player to sold in the details table. The trigger then invokes and automatically increase the player_count of the team to which the player is sold and reduce the balance of the team with the price of the player.

Timestamp Update Trigger

```
CREATE OR REPLACE TRIGGER players_updated_at_trigger
BEFORE UPDATE ON Players
FOR EACH ROW
BEGIN
    :NEW.updated_at := CURRENT_TIMESTAMP;
END;
/
```

```
SQL> CREATE OR REPLACE TRIGGER teams_audit_trigger
    AFTER INSERT OR UPDATE OR DELETE ON Teams -- Changed to AFTER for simplicity, ensures operation succeeded first
    FOR EACH ROW
    DECLARE...
Show more...
```

Audit Log Trigger

```
CREATE OR REPLACE TRIGGER teams_audit_trigger
AFTER INSERT OR UPDATE OR DELETE ON Teams
FOR EACH ROW
DECLARE
    v_old_data CLOB := NULL;
    v_new_data CLOB := NULL;
    v_action   VARCHAR2(10);
BEGIN
    IF UPDATING THEN
        v_action := 'UPDATE';

        IF :OLD.funds_remaining_auction2025 <>
        :NEW.funds_remaining_auction2025 OR (:OLD.funds_remaining_auction2025
        IS NULL AND :NEW.funds_remaining_auction2025 IS NOT NULL) OR
         (:OLD.funds_remaining_auction2025 IS NOT NULL AND :NEW.funds_remaining_auction2025 IS NULL)
         THEN
            v_old_data := '{"funds_remaining_auction2025": ' ||
            TO_CHAR(:OLD.funds_remaining_auction2025) || '}';
            v_new_data := '{"funds_remaining_auction2025": ' ||
            TO_CHAR(:NEW.funds_remaining_auction2025) || '}';
        END IF;
```

```
    ELSIF INSERTING THEN
        v_action := 'INSERT';

        v_new_data := '{"team_id": ' || TO_CHAR(:NEW.team_id) || ', "team_name": "'
        || :NEW.team_name || '"}'; -- Example

    ELSIF DELETING THEN
        v_action := 'DELETE';
        v_old_data := '{"team_id": ' || TO_CHAR(:OLD.team_id) || ', "team_name": "'
        || :OLD.team_name || '"}'; -- Example
    END IF;

    IF v_action = 'INSERT' OR v_action = 'DELETE' OR (v_action = 'UPDATE' AND
    (v_old_data IS NOT NULL OR v_new_data IS NOT NULL)) THEN
        INSERT INTO AuditLog(table_name, record_pk_value, action_type, admin_id, old_data, new_data)
        VALUES('Teams', CASE WHEN INSERTING THEN :NEW.team_id
         ELSE :OLD.team_id END, v_action, NULL,
              v_old_data,
              v_new_data);
    END IF;
```

```
END;
/
```

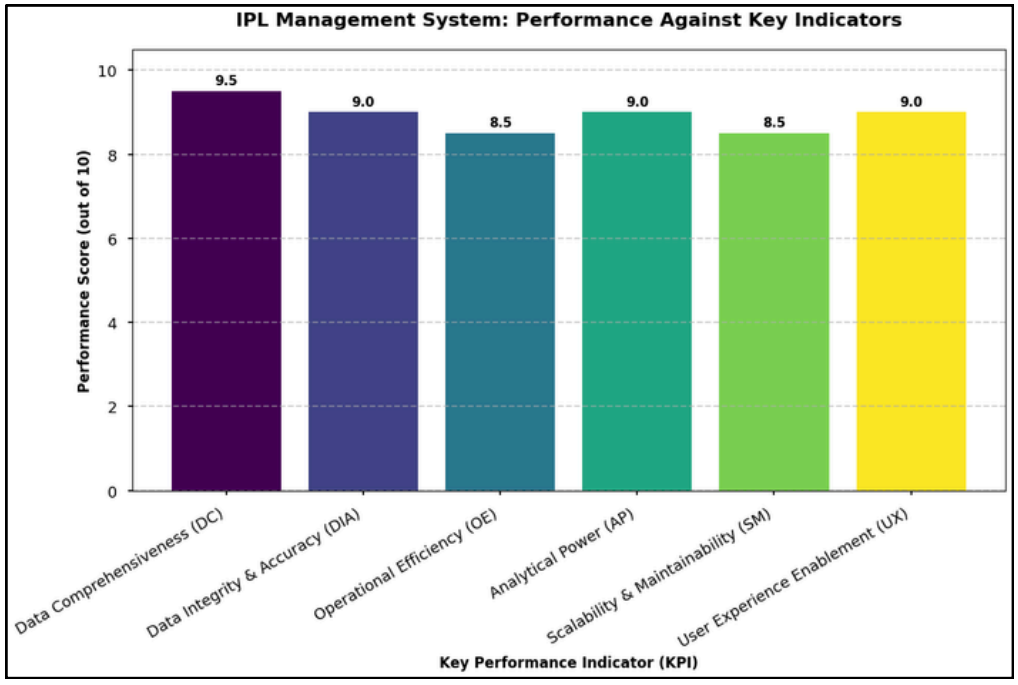
```
Trigger TEAMS_AUDIT_TRIGGER compiled

LINE/COL  ERROR
--------- -------------------------------------------------------------
26/10     PL/SQL: SQL Statement ignored
27/36     PL/SQL: ORA-00984: column not allowed here
Errors: check compiler log
Elapsed: 00:00:00.022
```

Statistical Analysis

Key Performance Indicators (KPIs) for an Ideal IPL Management System:

1. Data Comprehensiveness (DC): Covers all essential aspects of the IPL.
2. Data Integrity & Accuracy (DIA): Ensures data is correct, consistent, and reliable.
3. Operational Efficiency (OE): Streamlines management tasks and reduces manual effort.
4. Analytical Power (AP): Enables deep insights and data-driven decision-making.
5. Scalability & Maintainability (SM): Can grow with the league and is easy to update.
6. User Experience (UX): (Implied through backend design facilitating good frontend) Provides necessary data for a rich user interface for fans, admins, and teams.



Conclusion

In conclusion, the comprehensive IPL Management System designed and conceptualized within this project provides a robust, integrated, and highly effective solution for managing the intricate ecosystem of the Indian Premier League. Leveraging the power of SQL for data definition and querying, and PL/SQL for procedural automation including triggers and cursors, the system successfully centralizes and correlates a vast spectrum of data – encompassing detailed player profiles (auctioned, retained, unsold), complex auction results, team compositions, owner details, venue specifics, the full season schedule, dynamic points table standings, current season leaderboards, historical team statistics, all-time league records, and official announcements.

This project has demonstrated the critical role of a well-designed relational database and advanced SQL/PLSQL techniques in handling not just the auction, but the entire lifecycle and historical context of the IPL. The implementation facilitates seamless data storage, efficient retrieval, and sophisticated analysis far beyond basic auction metrics. Procedures and triggers automate crucial updates, such as points table calculations following match results or player contract synchronization post-auction, ensuring data integrity and operational efficiency. Advanced queries and cursor-based processes, like the head-to-head team comparison, unlock deeper analytical insights vital for teams, analysts, and broadcasters.

The system's architecture, adhering to sound normalization principles, ensures data accuracy, minimizes redundancy, and provides a scalable foundation capable of accommodating the league's evolution and growing data volumes across future seasons. By managing diverse data types – from player salaries and performance statistics to historical championship wins and ownership structures – within a single, cohesive framework, it eliminates the inefficiencies and error-proneness inherent in manual data management.

Overall, this advanced IPL Management System project powerfully showcases the capability of modern Database Management Systems, SQL, and PL/SQL to master the complexities of large-scale, data-intensive sports league operations. It stands as a testament to how these tools enable not only efficient management but also data-driven decision-making, enhanced stakeholder engagement (for teams, administrators, and fans alike), and the preservation of the league's rich history, thereby providing a solid and extensible foundation for future analytical applications and system enhancements.