COL362/632 Assignment-2: IPL Database Management

Deadline

The assignment is due on **18th Feb at 11:59 PM**. All submissions are to be made on Moodle.

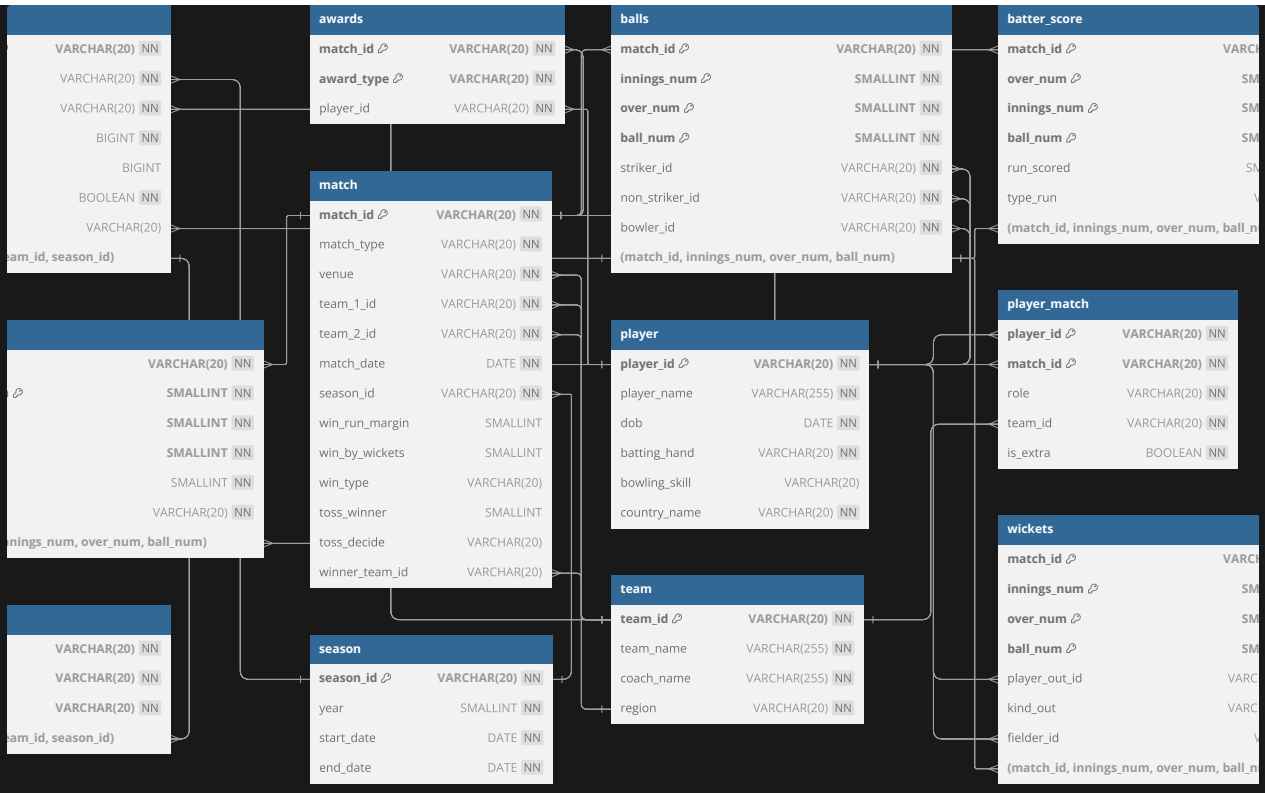
Objective

Your objective in this assignment is to design a database to manage IPL match data. As mentioned in the assignment, you must create tables with appropriate constraints(primary keys, foreign keys, unique, not null, etc) and additionally add triggers, views, functions, and rules that upload the database's integrity.

Tables

Table Name	Description
`auction`	Stores auction details for players, including base price, sold price, and team.
`awards`	Records awards given to players for specific matches.
`balls`	Tracks each ball bowled in a match, including striker, non-striker, and bowler.
`batter_score`	Records runs scored by batsmen for each ball.
`extras`	Tracks extra runs conceded in a match (e.g., wides, no-balls).
`match`	Stores match details, including teams, venue, date, and result.
`player`	Contains player details like name, date of birth, and skills.
`player_match`	Links players to matches and their roles in those matches.
`player_team`	Associates players with teams for specific seasons.
`season`	Stores information about cricket seasons, including start and end dates.
`team`	Contains team details like name, coach, and region.
`wickets`	Records wicket details for each ball, including the type of dismissal.

Schema



Tables Description

1. `auction`

Column Name	Data Type	Key	Description
`auction_id`	`varchar(20)`	Primary Key	Unique identifier for the auction.
`season_id`	`varchar(20)`	Foreign Key	References `season(season_id)`.
`player_id`	`varchar(20)`	Foreign Key	References `player(player_id)`.
`base_price`	`bigint`		Base price of the player.
`sold_price`	`bigint`		Price at which the player was sold.
`is_sold`	`boolean`		Indicates if the player was sold.
`team_id`	`varchar(20)`	Foreign Key	References `team(team_id)`.

---

2. `awards`

Column Name	Data Type	Key	Description
`match_id`	`varchar(20)`	Primary Key, Foreign Key	References `match(match_id)`.
`award_type`	`varchar(20)`	Primary Key	Name of the award
`player_id`	`varchar(20)`	Foreign Key	References `player(player_id)`.

---

3. `balls`

Column Name	Data Type	Key	Description
`match_id`	`varchar(20)`	Primary Key, Foreign Key	References `match(match_id)`
`innings_num`	`smallint`	Primary Key	Innings number.
`over_num`	`smallint`	Primary Key	Over number.
`ball_num`	`smallint`	Primary Key	Ball number.
`striker_id`	`varchar(20)`	Foreign Key	References `player(player_id)`
`non_striker_id`	`varchar(20)`	Foreign Key	References `player(player_id)`
`bowler_id`	`varchar(20)`	Foreign Key	References `player(player_id)`

---

4. `batter_score`

Column Name	Data Type	Key	Description
`match_id`	`varchar(20)`	Primary Key, Foreign Key	References `match(match_id)`
`over_num`	`smallint`	Primary Key	Over number.
`innings_num`	`smallint`	Primary Key	Innings number.
`ball_num`	`smallint`	Primary Key	Ball number.
`run_scored`	`smallint`		Runs scored on the ball.
`type_run`	`varchar(20)`		running or boundary

---

5. `extras`

Column Name	Data Type	Key	Description
`match_id`	`varchar(20)`	Primary Key, Foreign Key	References `match(match_id)`
`innings_num`	`smallint`	Primary Key	Innings number.
`over_num`	`smallint`	Primary Key	Over number.
`ball_num`	`smallint`	Primary Key	Ball number.
`extra_runs`	`smallint`		Extra runs conceded.
`extra_type`	`varchar(20)`		Type of extra (e.g., wide, no-ball).

---

6. `match`

Column Name	Data Type	Key	Description
`match_id`	`varchar(20)`	Primary Key	Unique identifier for the match.
`match_type`	`varchar(20)`		Type of match (league or playoff or knockout)
`venue`	`varchar(20)`	Foreign Key	Venue of the match. References `team(region)`
`team_1_id`	`varchar(20)`	Foreign Key	References `team(team_id)`.
`team_2_id`	`varchar(20)`	Foreign Key	References `team(team_id)`.
`match_date`	`date`		Date of the match.
`season_id`	`varchar(20)`	Foreign Key	References `season(season_id)`.
`win_run_margin`	`smallint`		Runs by which the match was won. (Defending)
`win_by_wickets`	`smallint`		Wickets by which the match was won.(Chasing)
`win_type`	`varchar(20)`		Type of win (e.g., runs, wickets).
`toss_winner`	`smallint`		Team that won the toss.
`toss_decide`	`varchar(20)`		Toss decision (bat or bowl).
`winner_team_id`	`varchar(20)`	Foreign Key	Team that won the match References `team(team_id)`.

---

7. `player`

Column Name	Data Type	Key	Description
`player_id`	`varchar(20)`	Primary Key	Unique identifier for the player.
`player_name`	`varchar(255)`		Name of the player.
`dob`	`date`		Date of birth of the player.
`batting_hand`	`varchar(20)`		Batting hand (left/right).
`bowling_skill`	`varchar(20)`		Bowling skill (e.g., fast, spin).
`country_name`	`varchar(20)`		Country of the player.

---

8. `player_match`

Column Name	Data Type	Key	Description
`player_id`	`varchar(20)`	Primary Key, Foreign Key	References `player(player_id)`.
`match_id`	`varchar(20)`	Primary Key, Foreign Key	References `match(match_id)`.
`role`	`varchar(20)`		Role of the player in the match.
`team_id`	`varchar(20)`	Foreign Key	References `team(team_id)`.
`is_extra`	`boolean`		Indicates if the player is an extra.

---

9. `player_team`

Column Name	Data Type	Key	Description
`player_id`	`varchar(20)`	Primary Key	Unique identifier for the Player
`team_id`	`varchar(20)`	Primary Key	Unique identifier for the Team
`season_id`	`varchar(20)`	Primary Key	Unique identifier for the season_id

---

10. `season`

Column Name	Data Type	Key	Description
`season_id`	`varchar(20)`	Primary Key	Unique identifier for the season.
`year`	`smallint`		Year of the season.
`start_date`	`date`		Start date of the season.
`end_date`	`date`		End date of the season.

---

11. `team`

Column Name	Data Type	Key	Description
`team_id`	`varchar(20)`	Primary Key	Unique identifier for the team.
`team_name`	`varchar(255)`		Name of the team.
`coach_name`	`varchar(255)`		Name of the coach.
`region`	`varchar(20)`		Region of the team.

---

12. `wickets`

Column Name	Data Type	Key	Description
`match_id`	`varchar(20)`	Primary Key, Foreign Key	References `match(match_id)`
`innings_num`	`smallint`	Primary Key	Innings number.
`over_num`	`smallint`	Primary Key	Over number.
`ball_num`	`smallint`	Primary Key	Ball number.
`player_out_id`	`varchar(20)`	Foreign Key	References `player(player_id)`.
`kind_out`	`varchar(20)`		Type of dismissal (e.g., caught, bowled).
`fielder_id`	`varchar(20)`	Foreign Key	References `player(player_id)`. Caught by pla

Unique Constraints

- **team**: team_name
- **team:** region
- **auction:** (player_id, team_id, season_id)

Note: If these constraints are violated during INSERT or UPDATE, postgres should return an error containing "**unique**" keyword.

Composite Foreign Key Relationships

- `fk_balls` in `extras` table:
  - **Foreign Keys**: `(match_id, innings_num, over_num, ball_num)`
  - **References**: `balls(match_id, innings_num, over_num, ball_num)`
- `fk_balls` in `wickets` table:
  - **Foreign Keys**: `(match_id, innings_num, over_num, ball_num)`
  - **References**: `balls(match_id, innings_num, over_num, ball_num)`
- `fk_balls` in `batter_score` table:
  - **Foreign Keys**: `(match_id, innings_num, over_num, ball_num)`
  - **References**: `balls(match_id, innings_num, over_num, ball_num)`
- `fk_auction` in `player_team` table:
  - **Foreign Keys:** `(player_id_id, team_id, season_id)`
  - **References:** `auction(player_id, team_id, season_id)`

Note: If these constraints are violated during INSERT or UPDATE, postgres should return an error containing "**foreign key**" keyword.

Allowed Values for Specific Columns

- **extras.extra_type**: no_ball, wide, byes, legbyes
- **awards.award_type**: orange_cap, purple_cap
- **batter_score.type_run:** running, boundary
- **match.match_type:** league, playoff, knockout
- **match.win_type:** runs, wickets, draw
  - match.win_type = runs implies the team who bat first have won
  - match.win_type = wickets implies the team who bowled first
  - match.win_type = draw imples a draw
- **match.toss_winner:** 1,2
  - match.toss_winner = 1 implies team_1 has won the toss
  - match.toss_winner = 2 implies team_2 has won the toss

- **match.toss_decide:** bowl, bat
  - match.toss_decide = bowl implies that toss winner team has chosen to bowl first
  - match.toss_decide = bat implies that toss winner team has chosen to bat first
- **player.batting_hand:** left, right
- **player.bowling_skill:** fast, medium, legspin, offspin
- **player_match.role:** batter, bowler, allrounder, wicketkeeper
- **wickets.kind_out:** bowled, caught, lbw, runout, stumped, hitwicket
- **batter_score**: `run_scored` should be greater than or equal to 0
- **extras**: `extra_runs` should be greater than equal to zero
- **player**: `dob` should be less than 1st Jan, 2016
- **season**: `year` should be between 1900 and 2025 (including both)
- **auction**: `base_price` should be greater than or equal to 1000000

Note: If these constraints are violated during INSERT or UPDATE, postgres should return an error containing "**check constraint**" keyword.

Single Table Constraints

Not NULL

- **auction**: auction_id, season_id, player_id, base_price, is_sold
- **awards**: match_id, award_type, player_id
- **balls**: match_id, innings_num, over_num, ball_num, striker_id, non_striker_id, bowler_id
- **batter_score**: match_id, over_num, innings_num, ball_num, run_scored
- **extras**: match_id, innings_num, over_num, ball_num, extra_runs, extra_type
- **match**: match_id, match_type, venue, team_1_id, team_2_id, match_date, season_id
- **player**: player_id, player_name, dob, batting_hand, country_name
- **player_match**: player_id, match_id, team_id, role, is_extra
- **player_team**: player_id, team_id, season_id

- **season**: season_id, year, start_date, end_date
- **team**: team_id, team_name, coach_name, region
- **wickets**: match_id, innings_num, over_num, ball_num, player_out_id, kind_out

Note: If these constraints are violated during INSERT or UPDATE, postgres should return an error containing "**null**" keyword.

Additional Not NULL

- **match** :
  - If the result(`win_type`) represents a draw then `win_run_margin` and `win_by_wickets` should be NULL.
  - If `win_type` does not represent a draw then exactly one of `win_run_margin` or `win_by_wickets` should be null.
  - If `win_type` is runs then `win_by_wickets` should be null and vice versa

  Note: If these constraints are violated during INSERT or UPDATE, postgres should return an error containing "**null**" keyword.
- **auction** :
  - If `is_sold` is true then `sold_price` and `team_id` should be not null and `sold_price` >= `base_price`

  Note: If these constraints are violated during INSERT or UPDATE, postgres should return an error containing "**null**" keyword.
- **wickets**
  - If `kind_out` is among caught, runout, stumped then `fielder_id` should be not null.

    Note: If these constraints are violated during INSERT or UPDATE, postgres should return an error containing "**null**" keyword.
  - If `kind_out` is "stumped" then `fielder_id` must have role of wicketkeeper in player_match.role

    Note: If these constraints are violated during INSERT or UPDATE, postgres should return an error containing "**for stumped dismissal, fielder must be a wicketkeeper**"

Advanced Table Constraints

Triggers

Automatic Insertion into `player_team` Table

Create a **trigger** on the `auction` table to automatically insert a corresponding record into the `player_team` table whenever an auction for a player is done (i.e. row is added in the `auction` table) and `is_sold` is true. The trigger should insert the `player_id`, `team_id`, and `season_id` from the `auction` table into the `player_team` table. This ensures that the `player_team` table remains consistent with auction results without requiring manual insertion.

Automatic season_id generation

`season_id` should comprise the prefix “IPL” along with the year, i.e. IPL{year}. Assume that every year only one IPL is played.

```
INSERT INTO public.season (year, start_date, end_date) VALUES (2025,
'2025-03-20', '2025-05-30');
```

The above query should add the following row

season_id	year	start_date	end_date
IPL2025	2025	2025-03-20	2025-05-30

Write a trigger for above.

match_id validation

`match_id` should be validated before insertion or updation in the table `match`. It should comprise of season_id and a serial number(3 digit). Format: season_id{SSS}.

The sequence number should start from 001 for each `season_id` and should increment by 1.

Create a trigger named `validate_match_id` to validate the `match_id`. If the `match_id` assigned to the match is valid then successfully insert the tuple in the table.

Example: (all columns are not shown)

Lets say the current `match` table looks like

match_id	season_id
IPL2024001	IPL2024
IPL2024002	IPL2024
IPL2025001	IPL2025

Valid insert statements(all column values are not shown):

```
INSERT INTO public.match ( match_id, season_id, ) VALUES ('IPL2023001', 'IPL2023'); INSERT INTO public.match ( match_id, season_id, ) VALUES ('IPL2024003', 'IPL2024'); INSERT INTO public.match ( match_id, season_id, ) VALUES ('IPL2025002', 'IPL2025');
```

Invalid insert statements(all column values are not shown):

```
INSERT INTO public.match ( match_id, season_id, ) VALUES ('IPL2023002', 'IPL2023'); INSERT INTO public.match ( match_id, season_id, ) VALUES ('IPL2024005', 'IPL2024'); INSERT INTO public.match ( match_id, season_id, ) VALUES ('IPL2024001', 'IPL2024'); INSERT INTO public.match ( match_id, season_id, ) VALUES ('IPL2025002', 'IPL2022');
```

Note: For invalid INSERT or UPDATE, postgres should return an error containing "**sequence of match id violated**".

Limit on International Players per Team

A team can have a **maximum of 3 international players**. A player is considered a **national player** if `player.country_name = 'India'`; otherwise, the player is classified as an **international player**.

Write a **trigger** to enforce this constraint.

Note: If this is violated during INSERT or UPDATE, postgres should return an error containing "**there could be atmost 3 international players per team per season**".

Limit on number of home matches

In each season, a team can play matches either on their home ground or an away ground. The constraints on venues are as follows:

1. **League Stage Matches:**
    - Each team can play **only one home match** and **one away match** against another team.

    Note: If number of league match condition is violated during INSERT or UPDATE, postgres should return an error containing "e**ach team can play only one home match in a league against another team**".
2. **Playoff and Knockout Matches:**
    - Each team can play any number of these matches. These matches can be played at either team's home ground or at a neutral venue.

A venue is considered a **"home" ground** for a team if the `match.venue` matches the `team.region`. It is considered an **"away" ground** if the `match.venue` does **not** match the `team.region`.

Write a **trigger** to enforce the above constraints when inserting records in the `match` table.

Note: If league match venue is not home ground of either team during INSERT or UPDATE, postgres should return an error containing "**league match must be played at home ground of one of the teams**".

Updating Rows

In the `match` table, when a match begins, its details are inserted with the following columns set to `NULL`:

- `win_run_margin`
- `win_by_wickets`
- `win_type`
- `toss_winner`
- `toss_decide`
- `winner_team_id`

The match record is then updated to include `toss_winner` and `toss_decide`. After that, entries for each delivery are inserted into the `balls` table. Finally, the `win_run_margin`, `win_by_wickets`, and `win_type` are updated to reflect the match outcome.

Your Task:

- **Automatically update the `winner_team_id`** in the `match` table based on the match result. If the match ends in a draw, set `winner_team_id` to `NULL`.
- **Insert two rows into the `awards` table** after the match concludes:
  1. One for the `orange_cap` (awarded to the player with the most runs).
  2. One for the `purple_cap` (awarded to the player with the most wickets).

In case of a tie (i.e., two players with the same number of runs or wickets), break the tie by selecting the player with the **lower `player_id`**.

Requirement:

Write a trigger that performs the above tasks automatically when the match result is finalized.

Deletion of Rows

In the schema, several rows are interconnected with foreign keys. Deleting one record from one table may affect more than one table.

Auction Deletion: When a player is sold in an auction (i.e., `is_sold = true`), and the corresponding record is deleted from the `auction` table, ensure that the following tables are also updated (i.e., rows related to that player and auction are deleted):

- `player_team`: Remove any records linking the player to a team in the season.

- `awards` : Remove any awards related to the player for the matches played in the season.
- `player_match` : Remove any records linking the player to a match in the season.
- `balls` , `batter_score` , `extras` , and `wickets` : Delete any rows associated with the player in these tables.

Match Deletion: If a match is deleted from the `match` table the following tables should also be updated:

- `awards` : Remove any referenced to the awards for that match
- `balls` : Delete all records related to that match.
- `batter_score` : Delete all score records for that match.
- `extras` : Delete all extra runs for that match.
- `wickets` : Delete all wicket records for that match.
- `player_match` : Remove any references to the match for players involved.

Season Deletion: If a season is deleted then the following tables should get affected:

- `auction` : Delete all the auctions that happened in that season
- `awards` : Delete all the awards given in that season
- `balls` : Delete all the ball played in all the matches in that season
- `batter_score` : Delete batter_score for that season
- `extras` : Delete extras for that season
- `match` : Delete matches played for that season
- `player_match` : Delete player_match entries for that season
- `player_team` : Delete player_team entries for that season
- `wickets` : Delete wickets taken during that season

Views

Maintain the following views.

batter_stats

column name	Description	datatype
player_id	player_id of the player	varchar(20)
Mat	Total matches played	smallint
Inns	Number of innings batted	smallint
R	Total runs scored	smallint
HS	Best individual score in an innings	smallint
Avg	Runs / Dismissals , 0 if no dismissals	double precision
SR	`(Runs / Balls Faced) × 100` , 0 if no balls faced	double precision
100s	100+ scores count in a match	smallint
50s	50–99 scores count in a match	smallint
Ducks	Number of times dismissed for 0	smallint
BF	Total balls faced	smallint
Boundaries	Total 4s & 6s hit	smallint
NO	Number of not-outs	smallint

If any delivery is in the `extras` table then it will not be considered as a ball faced by the batter.

bowler_stats

column name	Description	datatype