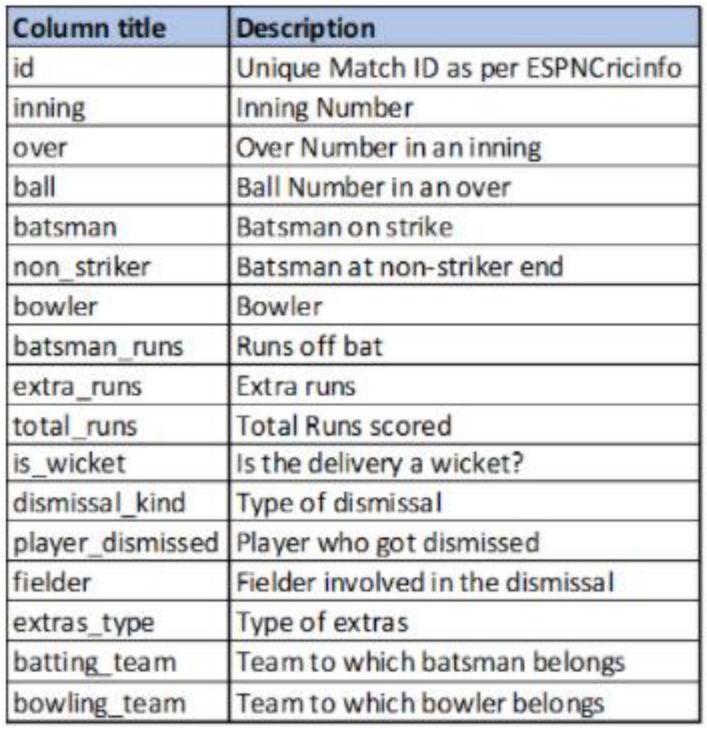
**Final Project - Sports Analytics using SQL**

In this project, you have to perform the job of a sports analyst. You are given two datasets related to IPL (Indian Premier League) cricket matches. One dataset contains ball-by-ball data and the other contains match-wise data. You have to import the datasets into an SQL database and perform the tasks given in this assignment to find important insights from this dataset.

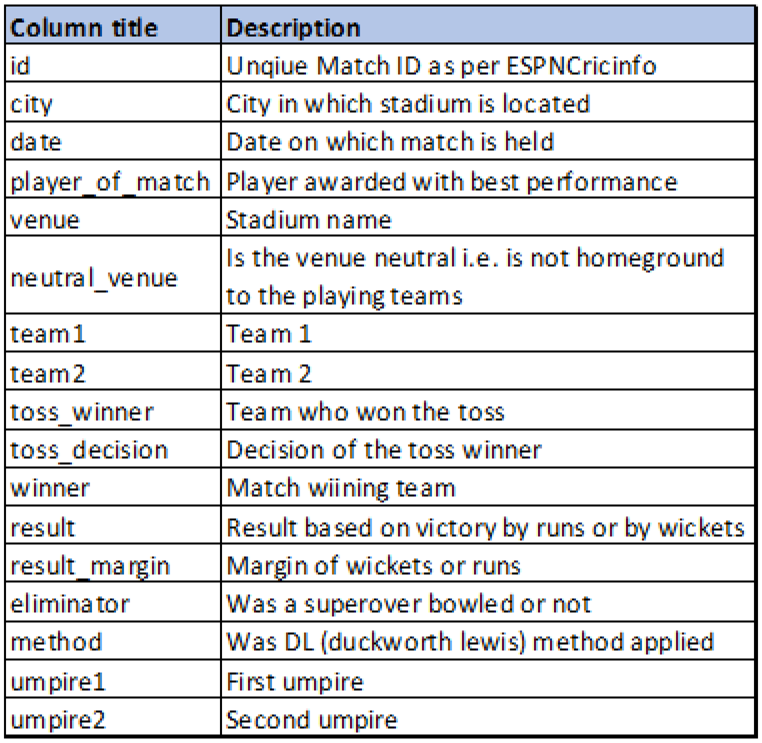
**About the Data**
Please download the datasets by clicking here and have them ready before we get started.

The first CSV file is for ball-by-ball data and it has information of all the 193468 balls bowled between the years 2008 and 2020. It has 17 columns and below is the details of those 17 columns:



| Column title | Description |
|---|---|
| id | Unique Match ID as per ESPNCricinfo |
| inning | Inning Number |
| over | Over Number in an inning |
| ball | Ball Number in an over |
| batsman | Batsman on strike |
| non_striker | Batsman at non-striker end |
| bowler | Bowler |
| batsman_runs | Runs off bat |
| extra_runs | Extra runs |
| total_runs | Total Runs scored |
| is_wicket | Is the delivery a wicket? |
| dismissal_kind | Type of dismissal |
| player_dismissed | Player who got dismissed |
| fielder | Fielder involved in the dismissal |
| extras_type | Type of extras |
| batting_team | Team to which batsman belongs |
| bowling_team | Team to which bowler belongs |

The second file contains match-wise data and has data of 816 IPL matches. This table has 17 columns and below is a short description of the columns in this table:



| Column title | Description |
|---|---|
| id | Unqiue Match ID as per ESPNCricinfo |
| city | City in which stadium is located |
| date | Date on which match is held |
| player_of_match | Player awarded with best performance |
| venue | Stadium name |
| neutral_venue | Is the venue neutral i.e. is not homeground to the playing teams |
| team1 | Team 1 |
| team2 | Team 2 |
| toss_winner | Team who won the toss |
| toss_decision | Decision of the toss winner |
| winner | Match wiining team |
| result | Result based on victory by runs or by wickets |
| result_margin | Margin of wickets or runs |
| eliminator | Was a superover bowled or not |
| method | Was DL (duckworth lewis) method applied |
| umpire1 | First umpire |
| umpire2 | Second umpire |

**Write queries for the following tasks:**

1. Create a table named ‘matches’ with appropriate data types for columns
2. Create a table named ‘deliveries’ with appropriate data types for columns
3. Import data from csv file ’IPL_matches.csv’ attached in resources to the table ‘matches’ which was created in Q1
4. Import data from csv file ’IPL_Ball.csv’ attached in resources to the table ‘deliveries’ which was created in Q2
5. Select the top 20 rows of the *deliveries* table after ordering them by id, inning, over, ball in ascending order.
6. Select the top 20 rows of the *matches* table.
7. Fetch data of all the matches played on 2nd May 2013 from the *matches* table..
8. Fetch data of all the matches where the result mode is ‘runs’ and margin of victory is more than 100 runs.
9. Fetch data of all the matches where the final scores of both teams tied and order it in descending order of the date.
10. Get the count of cities that have hosted an IPL match.
11. Create table *deliveries_v02* with all the columns of the table ‘*deliveries’* and an additional column *ball_result* containing values *boundary*, *dot* or *other* depending on the *total_run* (boundary for >= 4, dot for 0 and other for any other number)

(Hint 1 : CASE WHEN statement is used to get condition based results)
(Hint 2: To convert the output data of select statement into a table, you can use a subquery. Create table *table_name* as *[entire select statement].*

12. Write a query to fetch the total number of boundaries and dot balls from the *deliveries_v02* table.
13. Write a query to fetch the total number of boundaries scored by each team from the *deliveries_v02* table and order it in descending order of the number of boundaries scored.
14. Write a query to fetch the total number of dot balls bowled by each team and order it in descending order of the total number of dot balls bowled.
15. Write a query to fetch the total number of dismissals by dismissal kinds where dismissal kind is not NA
16. Write a query to get the top 5 bowlers who conceded maximum extra runs from the *deliveries* table
17. Write a query to create a table named *deliveries_v03* with all the columns of *deliveries_v02* table and two additional column (named *venue* and *match_date*) of *venue* and *date* from table *matches*
18. Write a query to fetch the total runs scored for each venue and order it in the descending order of total runs scored.
19. Write a query to fetch the year-wise total runs scored at *Eden Gardens* and order it in the descending order of total runs scored.
20. Get unique team1 names from the *matches* table, you will notice that there are two entries for *Rising Pune Supergiant* one with *Rising Pune Supergiant* and another one with *Rising Pune Supergiant****s***. Your task is to create a *matches_corrected* table with two additional columns *team1_corr* and *team2_corr* containing team names with replacing *Rising Pune Supergiant****s*** with *Rising Pune Supergiant*. Now analyse these newly created columns.
21. Create a new table deliveries_v04 with the first column as ball_id containing information of match_id, inning, over and ball separated by '-' (For ex. 335982-1-0-1 match_id-inning-over-ball) and rest of the columns same as deliveries_v03)
22. Compare the total count of rows and total count of distinct ball_id in deliveries_v04;
23. SQL Row_Number() function is used to sort and assign row numbers to data rows in the presence of multiple groups. For example, to identify the top 10 rows which have the highest order amount in each region, we can use row_number to

assign row numbers in each group (region) with any particular order (decreasing order of order amount) and then we can use this new column to apply filters.
Using this knowledge, solve the following exercise. You can use hints to create an additional column of row number.
Create table deliveries_v05 with all columns of deliveries_v04 and an additional column for row number partition over ball_id. (HINT : Syntax to add along with other columns,  row_number() over (partition by ball_id) as r_num)

24. Use the r_num created in deliveries_v05 to identify instances where ball_id is repeating. (HINT : select * from deliveries_v05 WHERE r_num=2;)
25. Use subqueries to fetch data of all the ball_id which are repeating. (HINT: SELECT * FROM deliveries_v05 WHERE ball_id in (select BALL_ID from deliveries_v05 WHERE r_num=2);