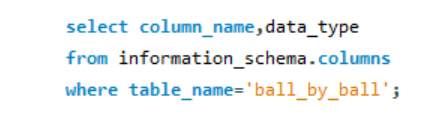
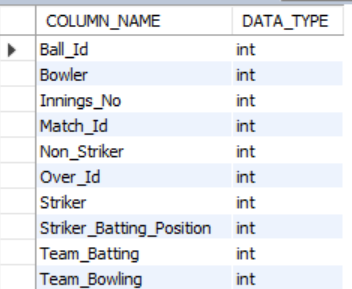
1. List the different data types of columns in table “ball\_by\_ball” (using information schema)

* Query:

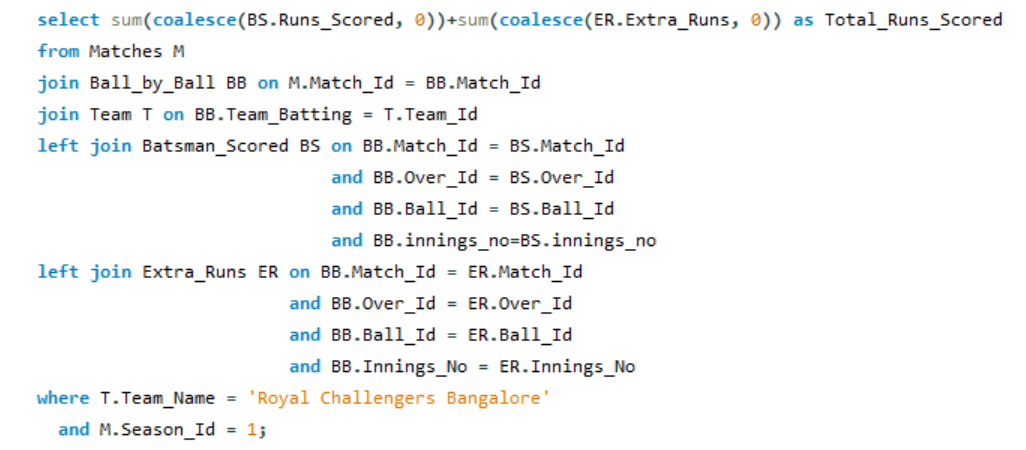


* Explanation:
* The query uses the information\_schema. columns view, which is part of the **information schema** in most SQL databases. This schema contains metadata about the database, including information about all tables, columns, constraints, etc.
* Result Output:



1. What is the total number of runs scored in 1st season by RCB (bonus: also include the extra runs using the extra runs table)

* Query



* Explanation:

This SQL query calculates the total runs scored by "Royal Challengers Bangalore" in Season 1. It sums the runs scored by batsmen (BS.Runs\_Scored) and extra runs (ER.Extra\_Runs) from various tables.

* From the Matches table (M): The query identifies the matches played.
* Join with Ball\_by\_Ball table (BB): Retrieves detailed ball-by-ball data for those matches.
* LEFT JOIN with Team table (T): Links batting team information to each ball bowled.
* LEFT JOIN with Batsman\_Scored table (BS): Gathers individual batsman runs, matching on match and ball details.
* LEFT JOIN with Extra\_Runs table (ER): Includes any extra runs awarded, ensuring accurate data collection.

The results are filtered to include only matches where "Royal Challengers Bangalore" was the batting team and from Season 1. The final output is labeled as Total\_Runs\_Scored.

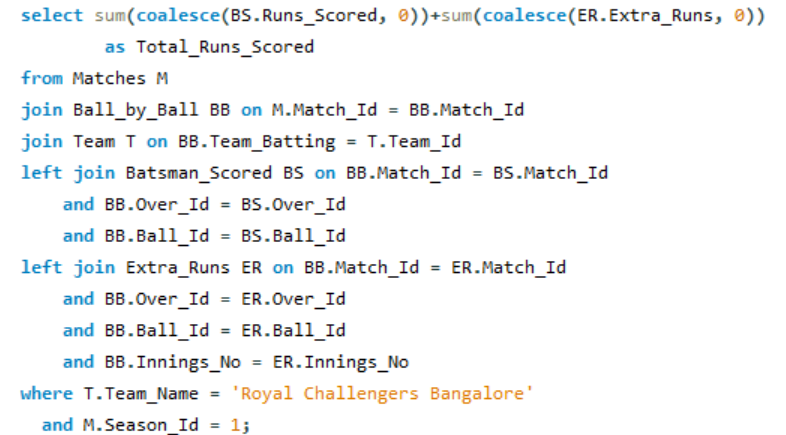
* Summary:
* The query sums up all the runs scored by the Royal Challengers Bangalore (RCB) in the first season, including both runs scored by the batsmen and any extra runs (such as wides, no-balls, leg-byes, etc.
* The **extra runs** are included in the calculation via the table, which is joined to the Ball\_by\_ball table. This accounts for runs not directly credited to the batsman but added to the team's score, like no-balls, wides, and byes.
* Result Output:



* The total of **1983 runs scored by RCB in the IPL season 1** including extra runs they got.

1. How many players were more than age of 25 during season 2 ?

* Query



* Explanation:

This SQL query counts the number of distinct players who are older than 25 years at the time of their first match in Season 2.

1. Selecting Count of Players: The outer query selects a count of distinct player\_id from the player table, which will give the total number of players meeting the criteria.
2. Joining Tables:
   * player\_match (pm): This join connects each player to their corresponding matches using player\_id.
   * matches (m): This join links match details to further filter based on the season.
3. Subquery for First Match Date:
   * The subquery calculates the minimum match date for Season 2, referred to as first\_match\_date.
4. Filtering Conditions:
   * The where clause restricts the results to matches in Season 2.
   * It checks if the age of the player (calculated using timestampdiff to find the difference in years between the player's date of birth (dob) and the first\_match\_date) is greater than 25 years.

The final output, labeled as players\_count, indicates the total number of players who meet these conditions.

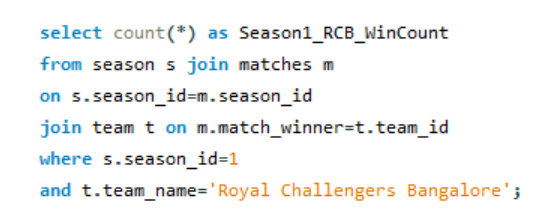
* Summary:
* **Purpose**: It calculates the total number of unique players who were older than 25 years during season 2.
* **Logic**: The query uses the TIMESTAMPDIFF function to calculate the players' ages at the time of the first match in season 2. It then counts the distinct players who were over 25 years old at that point.
* **First match date**: Calculating the age at the start of the season ensures that the players' age during the entire season is correctly assessed.
* Result output:



1. There are 82 players in the IPL season 2 with age above 25 years

1. How many matches did RCB win in season 1?

* Query



* Explanation:

This SQL query counts the number of matches won by the Royal Challengers Bangalore (RCB) during Season 1.

1. Selecting Win Count: The outer query selects a count of all rows that meet the specified criteria, labeling the result as Season1\_RCB\_WinCount.
2. Joining Tables:
   * season (s): This table contains information about the different seasons. It is joined to the matches table using the season\_id to link the season with its respective matches.
   * matches (m): This table records the match details, including the winner of each match.
   * team (t): This table contains team information, which is joined to the matches table to identify the winning team.
3. Filtering Conditions:
   * The where clause specifies that the query should only consider matches from Season 1 (s.season\_id=1).
   * It further filters the results to include only those matches where the winning team is the Royal Challengers Bangalore (t.team\_name='Royal Challengers Bangalore').

The final output provides the total count of matches won by RCB in Season 1, helping analyze their performance during that season.

* Summary:

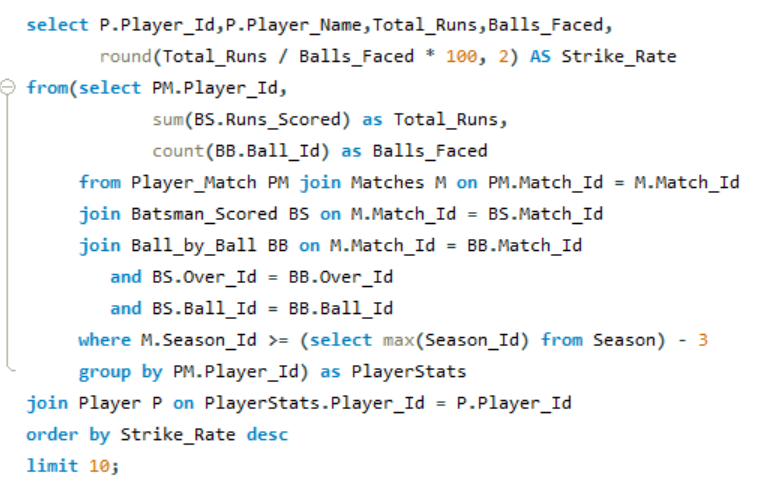
1. The query looks for matches in season 1 where the team\_id of the match\_winner corresponds to RCB, and then counts how many such matches exist.

* Result



1. RCB have won total 4 matches in the IPL Season 1.
2. List top 10 players according to their strike rate in last 4 seasons

* Query



* Explanation:

This SQL query retrieves the top 10 players based on their strike rate over the last four seasons.

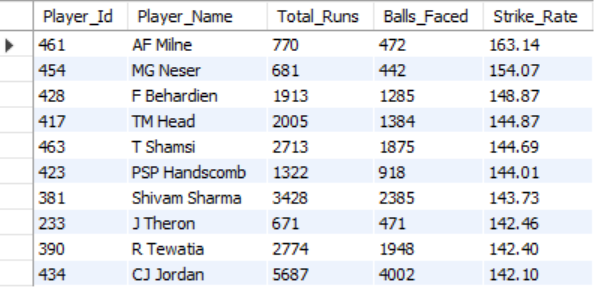
1. Inner Query (PlayerStats):
   * The inner query calculates two key statistics for each player:
     + Total\_Runs: The total runs scored by each player across matches.
     + Balls\_Faced: The total number of balls faced by each player in those matches.
   * Joins:
     + The Player\_Match table is joined with the Matches table to link players to their respective matches.
     + The Batsman\_Scored table is joined to capture the runs scored in those matches.
     + The Ball\_by\_Ball table is joined to count the balls faced by the players.
   * Filtering Condition:
     + The matches considered are restricted to those played in the last four seasons (M.Season\_Id >= (select max(Season\_Id) from Season) - 4).
   * Aggregation:
     + The results are grouped by PM.Player\_Id to aggregate the total runs and balls faced for each player.
2. Outer Query:
   * This part of the query retrieves player details from the Player table, joining it with the results of the inner query (PlayerStats).
   * It calculates the Strike\_Rate for each player as (Total\_Runs / Balls\_Faced \* 100), rounding to two decimal places.
   * The output columns include Player\_Id, Player\_Name, Total\_Runs, Balls\_Faced, and Strike\_Rate.
3. Ordering and Limiting Results:
   * The results are ordered in descending order based on the Strike\_Rate.
   * Finally, the query limits the results to the top 10 players.

This query effectively highlights the top-performing batsmen based on their ability to score runs efficiently over recent seasons, providing insights into their batting performance.

* Summary:

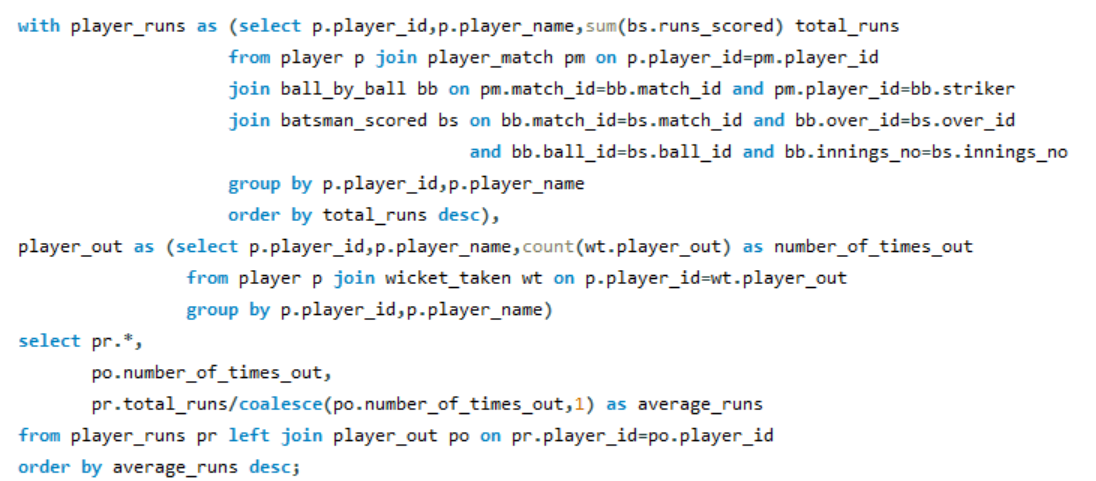
1. For each player, it adds up all the runs they scored and counts the total balls they faced in the last four seasons.

* Strike rate is calculated as (Total Runs / Balls Faced) \* 100. This tells us how many runs a player score on average per 100 balls faced.
* The query then lists the top 10 players with the highest strike rates, including their name, total runs, balls faced, and strike rate, in order from highest to lowest strike rate.
* Result output



1. What is the average runs scored by each batsman considering all the seasons?

* Query:



* Explanation:

This SQL query calculates the average runs scored by each batsman and retrieves the top 10 batsmen based on their average runs.

1. **CTE: player\_runs**:
   * This part calculates the **total runs** scored by each player.
   * It joins several tables:
     + **player**: Contains player information.
     + **player\_match**: Links players to the matches they played.
     + **ball\_by\_ball**: Contains details about each ball bowled in the matches.
     + **batsman\_scored**: Records runs scored in each match.
   * The results are grouped by player ID and name, summing their runs and ordering them from highest to lowest.
2. **CTE: player\_out**:
   * This part counts how many times each player has been out.
   * It joins the **player** table with the **wicket\_taken** table, which records when players are out.
   * The results are grouped by player ID and name to get the total number of outs for each player.
3. **Final Selection**:
   * The final part of the query combines results from player\_runs and player\_out.
   * It includes the total runs, number of times out, and calculates the **average runs** scored by dividing total runs by the number of times out (using COALESCE to avoid division by zero).
   * The results are ordered by average runs in descending order, and only the top 10 players are returned.

In summary, this query provides a list of the top 10 batsmen based on their average runs scored, taking into account their performance in matches and how often they get out.

* Summary:

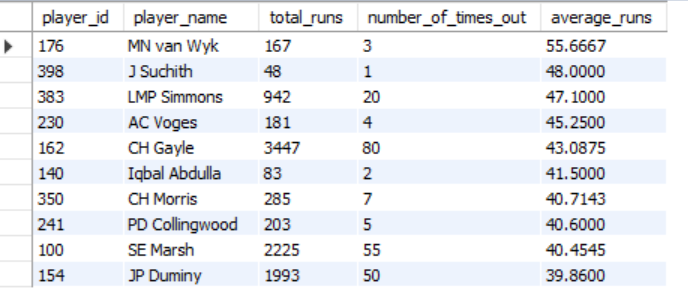
The output will provide a list of each batsman along with:

* Their total runs scored.
* The number of times they have been out.
* Their average runs scored per dismissal, allowing for an assessment of batting efficiency across all seasons.

In summary, this query provides a list of the top 10 batsmen based on their average runs scored, taking into account their performance in matches and how often they get out.

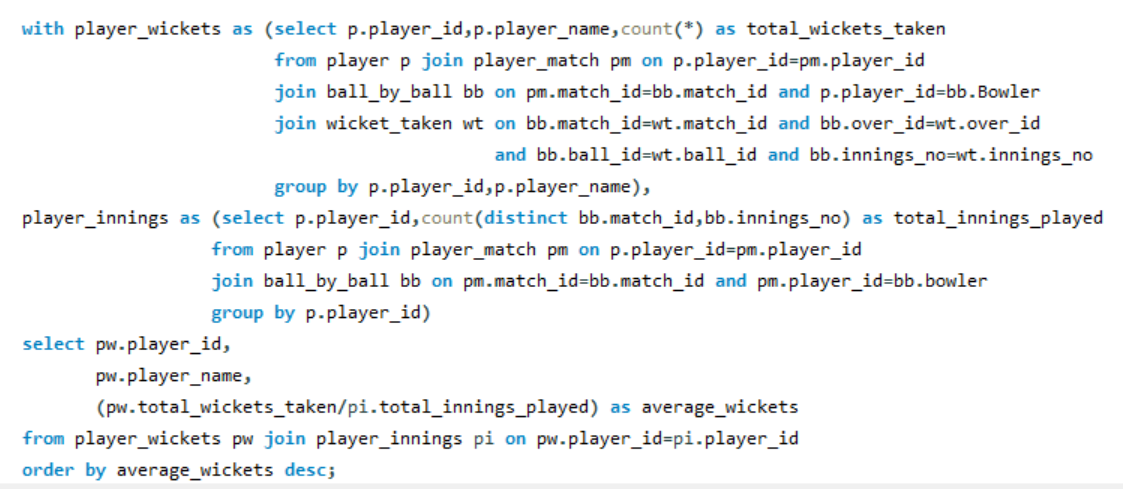
* Result output:

1. These are top 10 player from the above query who is having highest batting average considering all the season.



1. What are the average wickets taken by each bowler considering all the seasons?

* Query:



* Explanation:
* This SQL query calculates the average wickets taken by each bowler across all seasons and returns the results in descending order.

1. **CTE: player\_wickets**:
   * This part counts the **total wickets taken** by each player (bowler).
   * It joins the following tables:
     + **player**: Contains information about the players.
     + **player\_match**: Links players to the matches they played.
     + **ball\_by\_ball**: Contains details about each ball bowled, specifically identifying the bowler.
     + **wicket\_taken**: Records the details of wickets taken in each match.
   * The results are grouped by player ID and name, calculating the total wickets taken by each bowler.
2. **CTE: player\_innings**:
   * This part calculates the **total innings played** by each bowler.
   * It joins the **player** table with **player\_match** and **ball\_by\_ball** tables.
   * The query counts distinct matches and innings for each bowler to get the total innings they have bowled.
   * Results are grouped by player ID.
3. **Final Selection**:
   * The final part of the query retrieves the player ID, player name, and calculates the **average wickets** by dividing the total wickets taken by the total innings played.
   * It uses a **JOIN** to combine results from player\_wickets and player\_innings based on player ID.
   * The results are ordered by average wickets in descending order.

* Summary:

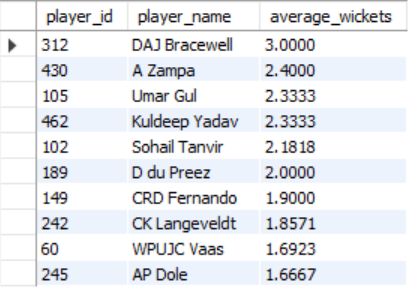
The output will provide a list of each bowler, including:

* Their total wickets taken.
* The number of innings played.
* Their average wickets taken per innings, helping to identify the most effective bowlers in the dataset.

This query effectively gives a comprehensive view of bowling performance, highlighting the bowlers who are most successful relative to the number of innings they've bowled.

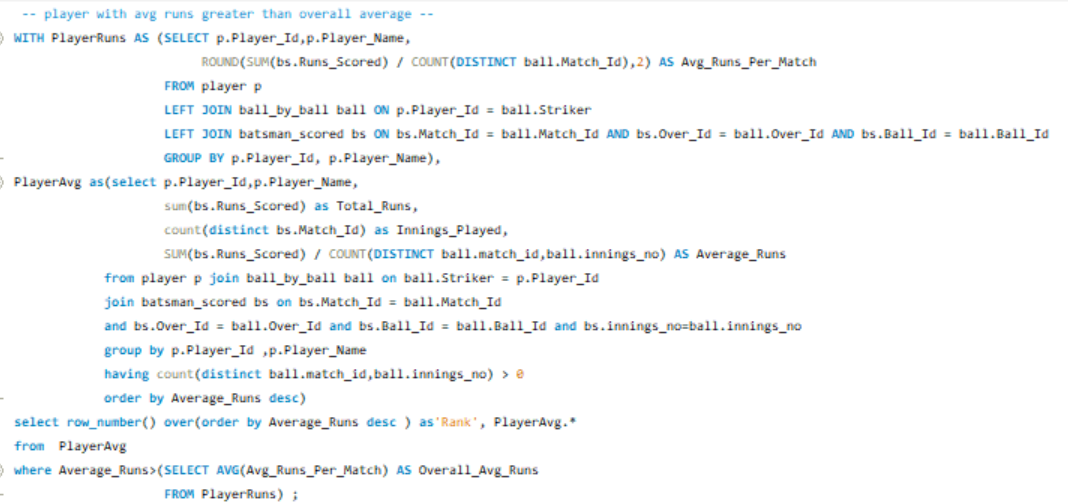
* Result Output:

1. These are top 10 players who is having highest wicket average per innings.

****

1. List all the players who have average runs scored greater than overall average and who have taken wickets greater than overall average

* Query: player runs greater than the overall average runs



* Explanation:

This SQL query identifies players who have an average runs scored per match that is higher than the overall average runs scored by all players.

1. Calculate Average Runs per Match (PlayerRuns CTE):
   * This part computes each player's average runs by summing their total runs and dividing by the number of matches they played.
   * It uses data from the player, ball\_by\_ball, and batsman\_scored tables.
2. Calculate Total Runs and Average Runs (PlayerAvg CTE):
   * This section calculates each player's total runs and their average runs per innings.
   * It also uses the same tables and counts the innings played.
3. Select Players with Above-Average Performance:
   * The final selection ranks players based on their average runs and filters those who exceed the overall average runs calculated in the first step.

* Summary:

1. The query lists players whose average runs scored per match are above the overall average, allowing for the identification of top-performing batsmen.

* Query: Player wickets greater the overall average wickets



* Explanation:

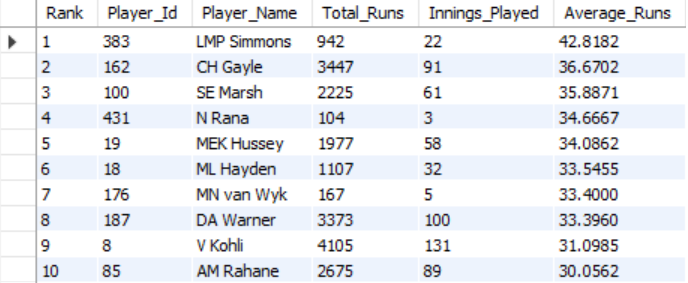
This SQL query identifies bowlers whose average wickets taken per match are higher than the overall average wickets taken by all bowlers.

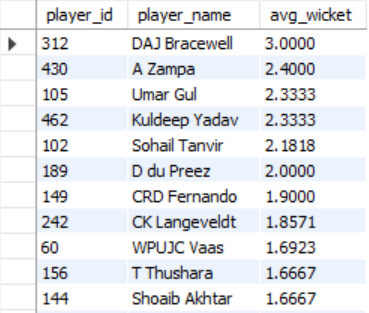
1. Calculate Total Wickets per Player (player\_wickets CTE):
   * This part counts the total wickets taken by each player.
   * It joins the player, player\_match, ball\_by\_ball, and wicket\_taken tables to gather the necessary data.
2. Count Matches Played by Each Player (played\_match CTE):
   * This section counts how many matches each player has played.
   * It uses the player and player\_match tables for this purpose.
3. Calculate Average Wickets per Player (avg\_wickets CTE):
   * Here, the average wickets taken per match for each player are calculated by dividing their total wickets by the number of matches they played.
   * It joins the results from the previous two CTEs.
4. Select Players with Above-Average Wickets:
   * The final selection filters for players whose average wickets are greater than the overall average wickets computed in the previous step.

* Summary:

1. The query lists bowlers whose average wickets taken per match exceed the overall average, highlighting the most effective bowlers in the dataset.

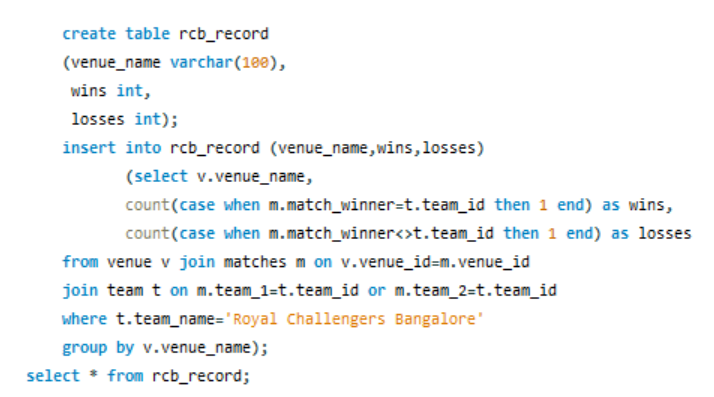
* Output:





1. Create a table RCB\_RECORD table that shows wins and losses of RCB in an individual venue.

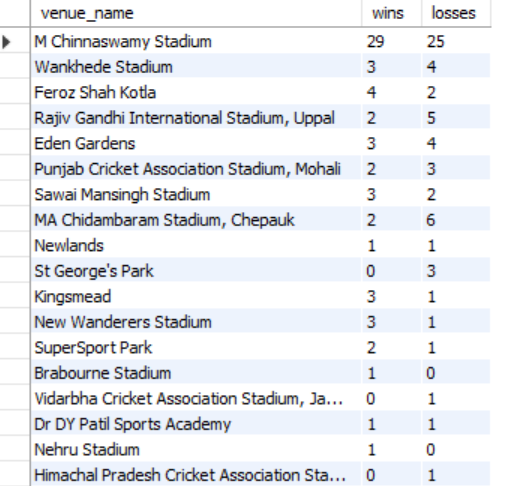
* Query:



* Explanation:
* **Create the Table**:
* A table named rcb\_record is created with three columns: venue\_name, wins, and losses.
* **Insert Data**:
* The table is filled with data showing how many matches the Royal Challengers Bangalore (RCB) has won and lost at each venue.
* It counts wins and losses by checking the match results where RCB is involved.
* **Retrieve Data**:
* Summary:

1. The SQL script creates a table called rcb\_record to show the Royal Challengers Bangalore's (RCB) wins and losses at different venues. It counts how many matches RCB won and lost at each venue and stores this information in the table. Finally, it retrieves and displays all the records from the rcb\_record table.

* Output:



1. What is the impact of bowling style on wickets taken.

* General Explanation:

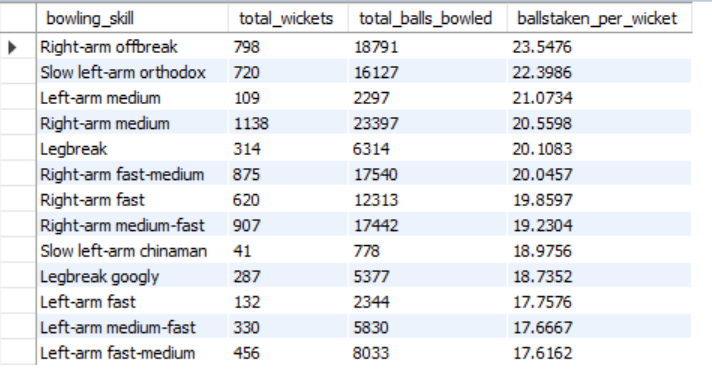
 Objective: Determine how different bowling styles influence the number of wickets taken.

 Hypothesis: Certain bowling styles (e.g., spin bowling) are more effective in taking wickets compared to others (e.g., medium pace) under specific conditions.

* Query:



* Explanation:
* Purpose: The query aims to determine how effective different bowling styles are at taking wickets in cricket.
* Common Table Expressions (CTEs):
* bowling\_skill\_wicket: This part counts how many wickets each bowling style has taken. It gathers data from several tables to connect players, their matches, the balls they bowled, and the wickets taken.
* bowling\_skill\_ball: This part counts the total number of balls bowled by each bowling style, again gathering data from related tables.
* Final Output:
* The main query combines the two CTEs to produce:
  + The bowling skill type.
  + Total wickets taken by that bowling style.
  + Total balls bowled by that bowling style.
  + A calculated ratio of balls bowled per wicket taken (lower values mean better efficiency).
* Results: The output is ordered to show which bowling styles are the most effective at taking wickets, based on how many balls they bowl for each wicket.
* Output:



* Additional general points:

 **Bowling Styles**: The primary bowling styles include fast bowling, medium pace, swing bowling, spin bowling, and seam bowling. Each style has distinct techniques and strategies that affect how they take wickets.

 **Match Conditions**: The effectiveness of a bowling style often depends on the match conditions, such as pitch type, weather, and humidity. For example, spin bowlers may thrive on dry, turning pitches, while fast bowlers may excel in overcast conditions that help the ball swing.

 **Wicket Type**: Certain bowling styles are more effective on specific types of wickets. Fast bowlers can exploit bouncy pitches, while spinners may benefit from turning tracks. The surface condition significantly influences the outcome of the bowling style's effectiveness.

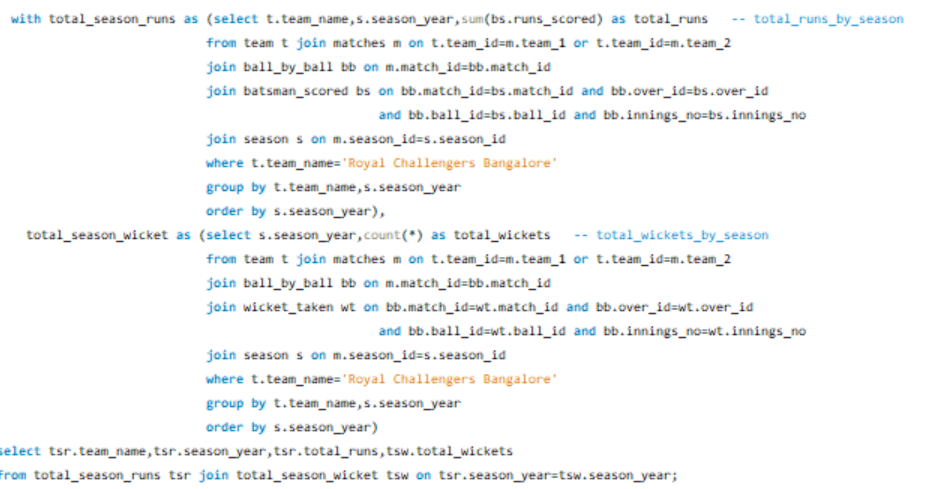
 **Batsman’s Weaknesses**: Different batsmen have varying strengths and weaknesses against specific bowling styles. A bowler's ability to exploit a batsman's weakness—such as their struggle against spin or seam movement—can lead to increased wicket-taking opportunities.

 **Variation in Delivery**: Bowlers who can vary their pace, spin, or seam movement within their bowling style are often more successful. Variations such as Yorkers, bouncers, and slower balls can disrupt a batsman's timing and lead to dismissals.

 **Field Placement**: Different bowling styles require specific field placements. For instance, spinners may need more catching positions, while fast bowlers may have a slip cordon. Effective field placements can enhance wicket-taking opportunities.

1. Write the sql query to provide a status of whether the performance of the team better than the previous year performance on the basis of number of runs scored by the team in the season and number of wickets taken

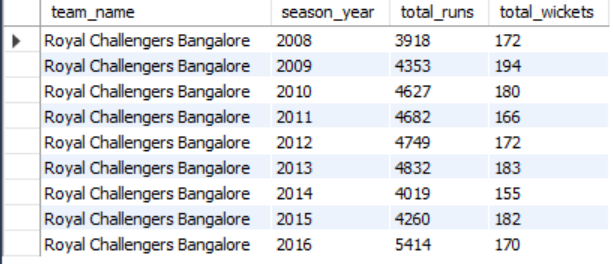
* Query:



* Explanation:
* **Total Runs Calculation**:
* The first Common Table Expression (CTE), total\_season\_runs, calculates the total runs scored by the Royal Challengers Bangalore (RCB) for each season.
* It joins the team, matches, ball\_by\_ball, batsman\_scored, and season tables to gather data on runs scored.
* The results are grouped by team name and season year.
* **Total Wickets Calculation**:
* The second CTE, total\_season\_wicket, counts the total wickets taken by RCB in each season.
* Similar to the first CTE, it joins the necessary tables to extract wicket data and groups the results by team name and season year.
* **Final Selection**:
* The main query selects the team name, season year, total runs, and total wickets by joining the two CTEs on the season year.
* Summary:

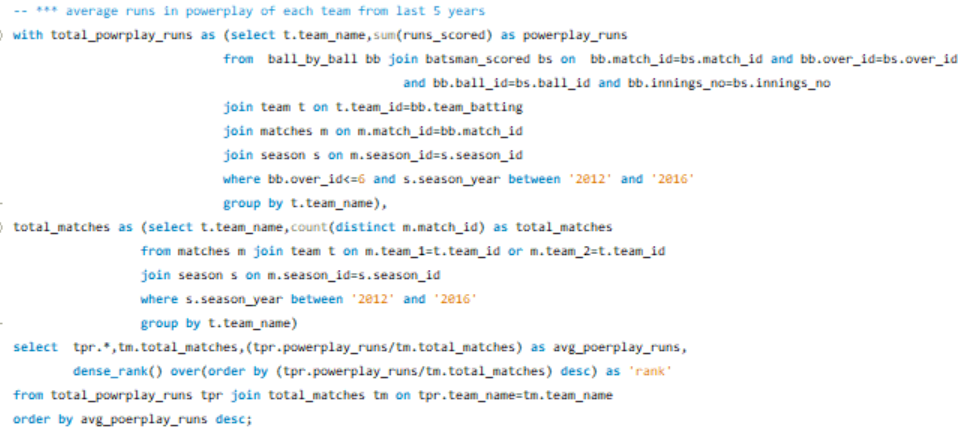
1. This SQL query retrieves the performance statistics (total runs scored and total wickets taken) of the Royal Challengers Bangalore for each season. It provides insights into the team's performance over the years, allowing for comparisons to determine if their performance has improved compared to previous seasons.

* Output:



1. Can you derive more KPIs for the team strategy if possible?

* Additional **Key Performance Indicator**:
* Focusing on powerplay performance of the team
* Focusing on death over performance of the team
* Powerplay performance in batting:
* Query:



* Explanation:

 **Calculate Powerplay Runs:** The total\_powrplay\_runs CTE (Common Table Expression) sums up the runs scored by each team in the first 6 overs (powerplay) from 2012 to 2016.

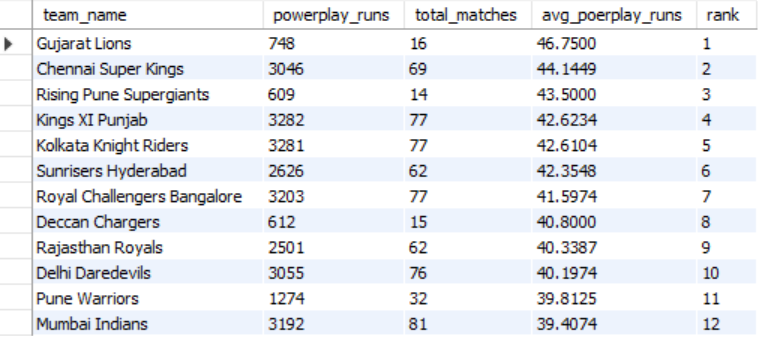
 **Count Total Matches:** The total\_matches CTE counts the distinct number of matches each team played during these years.

 **Calculate Average and Rank:** In the main query, it calculates the average powerplay runs per match for each team (powerplay\_runs/total\_matches). It ranks teams based on this average, from highest to lowest

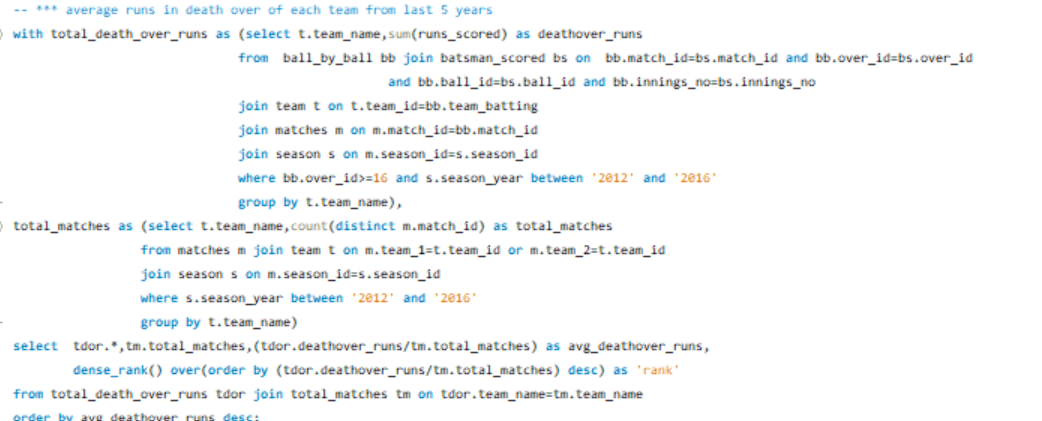
* Summary:

1. The final output lists each team, their total matches, average powerplay runs, and rank, ordered by the average powerplay runs in descending order.

* Output:



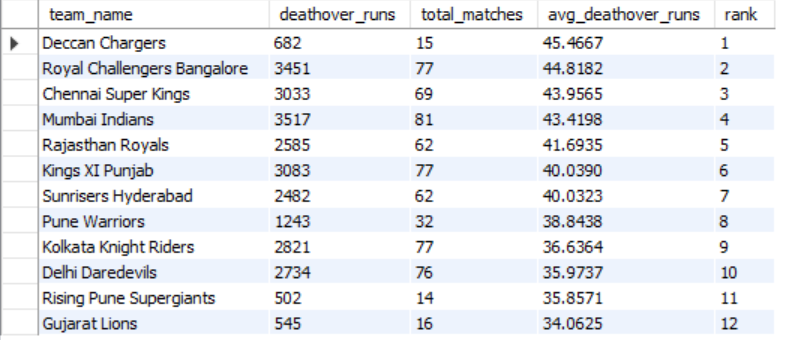
* Death over performance in batting
* Query:



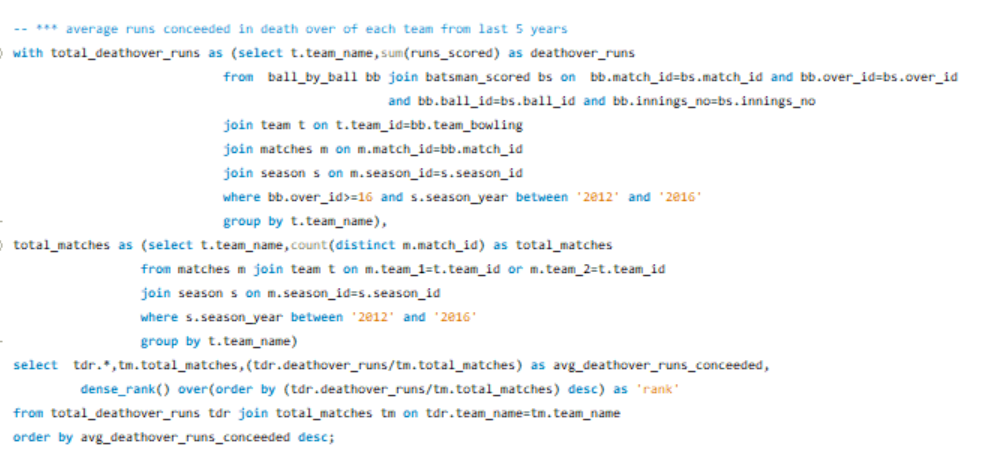
* Explanation:
* This query calculates the average runs scored by each team in the death overs (last 4 overs, 16–20) over the five-year period from 2012 to 2016 and ranks teams based on this average.
* Calculate Death Over Runs: The total\_death\_over\_runs CTE sums up the runs scored by each team in the death overs (overs 16 to 20) for matches played from 2012 to 2016.
* Count Total Matches: The total\_matches CTE counts the number of distinct matches each team played during the same time period.
* Calculate Average and Rank: In the main query, it calculates each team's average death-over runs per match (deathover\_runs/total\_matches) and ranks teams based on this average from highest to lowest.
* Summary:

1. The final output lists each team with their total matches, average death-over runs, and rank, ordered by the average death-over runs in descending order.

* Output:



* Death over performance in bowling -Runs conceded
* Query:



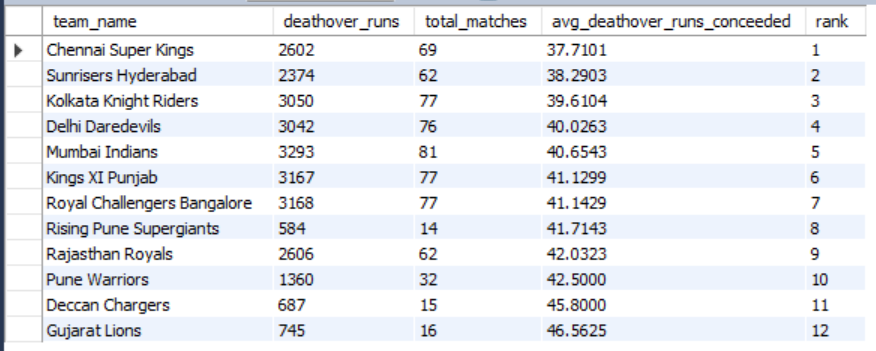
* Explanation:
* This query calculates the average runs conceded by each team in the death overs (overs 16 to 20) over a five-year period (2012 to 2016), ranking the teams based on this average.

Here’s an outline of the process:

* Calculate Death Over Runs Conceded: The total\_deathover\_runs CTE sums the runs conceded by each team's bowlers in the death overs (overs 16–20) across all matches played from 2012 to 2016.
* Count Total Matches Played: The total\_matches CTE counts the number of matches each team participated in during the same period, regardless of whether they were the bowling or batting team.
* Calculate Average and Rank: In the main query, it calculates each team's average death-over runs conceded per match (deathover\_runs/total\_matches) and ranks teams in descending order based on this average.
* Summary:

1. The final output includes each team’s name, total matches played, average death-over runs conceded, and their rank, with teams ranked from the highest to the lowest average runs conceded in the death overs.

* Output:



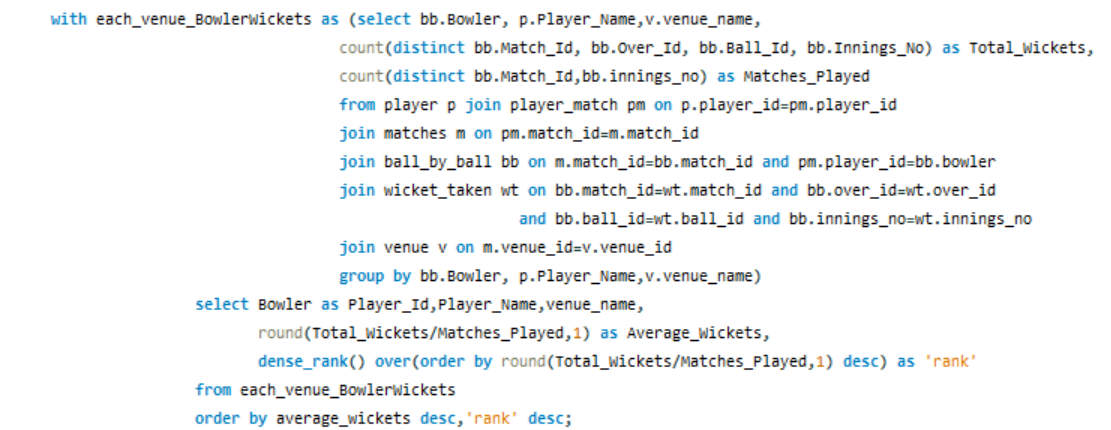
* Overall study:

1. Compare to any other team Gujarat lions’ team has highest average in powerplay runs

This will help team to score more runs.

1. In death overs RCB having very good average score 44.8 runs next to deccan charges compare to any other team.
2. In bowling controlling opponent in death over (not conceded more runs) is an important strategy to minimize opponent score.
3. CSK in the top team in death over with average runs conceded 37.7 followed by SRH with 38.3 and RCB is very low performance compare to CSK and SRH with average runs conceded is 41.1
4. In the overall study RCB need to improve in the death over in bowling and batting side of RCB is quite good at both powerplay and death overs.
5. Using SQL, write a query to find out average wickets taken by each bowler in each venue. Also rank the gender according to the average value.

* Query:



* Explanation:

 Common Table Expression (CTE): The query starts with a CTE named EachVenueBowlerWickets. This section gathers the necessary data:

* It selects the bowler's ID (bb.Bowler), the bowler's name (p.Player\_Name), and the venue name (v.Venue\_Name).
* It counts distinct instances of wickets taken to get the total wickets (Total\_Wickets).
* It also counts distinct matches played to get the total matches (Matches\_Played).

 Joining Tables:

* The query joins several tables:
  + Player, Player\_Match, Matches, Ball\_By\_Ball, Wicket\_Taken, and Venue.
* These joins ensure the data from each table is correctly related based on player and match IDs.

 Grouping Data:

* The data is grouped by bowler, player name, and venue name to aggregate the total wickets and matches for each bowler at each venue.

 Main Query: The main query selects data from the CTE:

* It retrieves the bowler's ID, name, venue name, and calculates the average wickets by dividing total wickets by matches played, rounding it to one decimal place (Average\_Wickets).
* It uses DENSE\_RANK() to rank the bowlers based on their average wickets in descending order.

 Ordering the Result:

* Finally, the results are ordered first by Average\_Wickets in descending order and then by Rank.
* Summary:

 **Purpose:** The query calculates the average wickets for each bowler at different venues and ranks them by their averages.

 **Key Functions:** It uses aggregation functions like COUNT and ROUND, along with ranking functions like DENSE\_RANK().

 **Output:** The result provides the bowler's ID, name, venue, average wickets, and rank based on average performance.

* Output:

\*Example out put is just top players based on average. Actual output is too big.

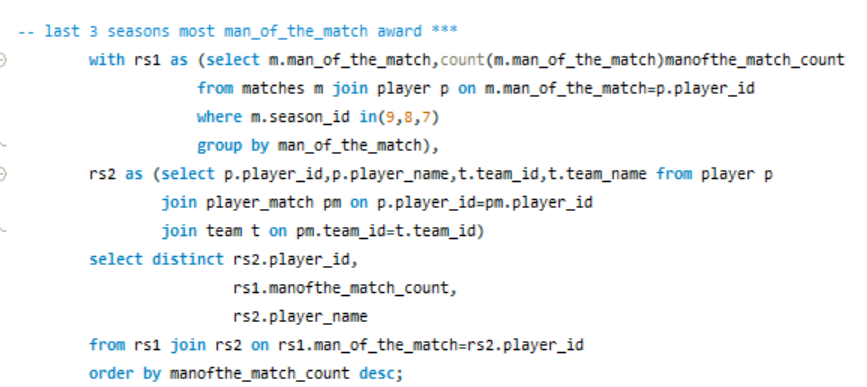


1. Which of the given players have consistently performed well in past seasons? (will you use any visualisation to solve the problem)

* To get the consistently well performed players the following analysis and KPI have used.

1. last 3 seasons most man\_of\_the\_match award \*
2. players in the list of top 5 scorer in last 3 seasons \*
3. top 10 bowlers from last each 3 seasons\*

* Query: -last 3 seasons most man\_of\_the\_match award



* Explanation:

 Count Man of the Match Awards:

* The first CTE (rs1) counts the number of "Man of the Match" awards each player received in the specified seasons. It joins the matches table with the player table to get the player IDs associated with the awards and groups the results by player ID.

 Retrieve Player and Team Information:

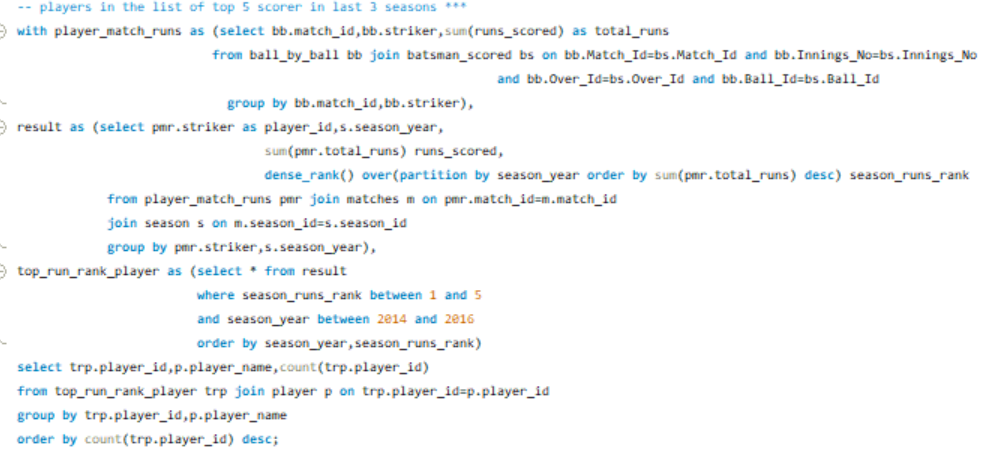
* The second CTE (rs2) gathers player details, including their names and associated team names, by joining the player, player\_match, and team tables.

 Join and Select Distinct Players:

* The main query joins the two CTEs (rs1 and rs2) on the player ID to associate each player with their award count. It selects distinct player IDs, their award counts, and names.

 Order the Results:

* The results are ordered by the count of "Man of the Match" awards in descending order, showing the players with the most awards at the top.
* Summary:
* The final output will provide a list of players, the number of "Man of the Match" awards they received, and their corresponding player names, sorted by the awards count.
* Query: - players in the list of top 5 scorer in last 3 seasons

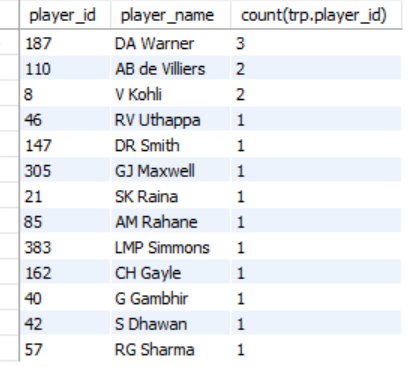


* Explanation:
* This query identifies players who ranked in the top 5 scorers in each of the last three seasons (from 2014 to 2016) and counts how many times each player appeared in this top 5 list across those seasons.

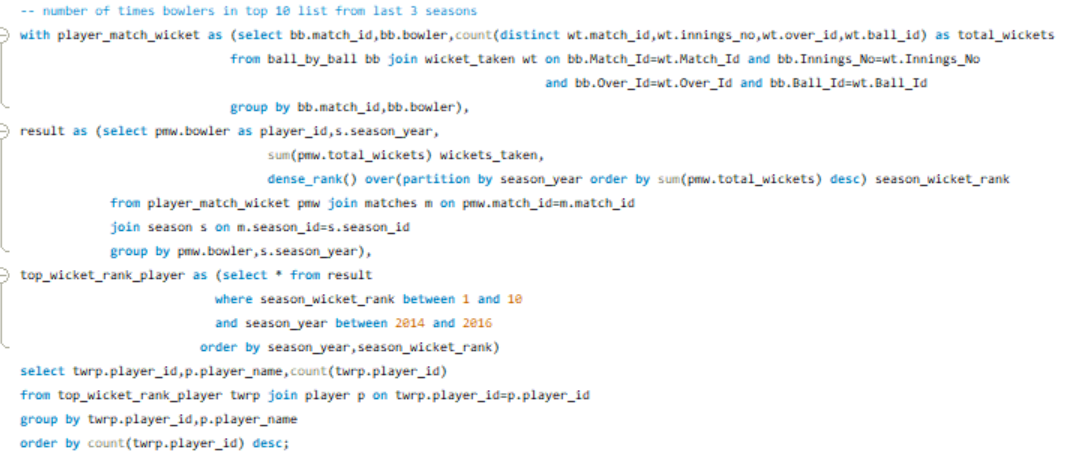
Here’s a breakdown of the steps involved:

1. Calculate Total Runs per Match:
   * The player\_match\_runs CTE sums up the runs scored by each player (striker) in each match. It joins the ball\_by\_ball and batsman\_scored tables to gather the necessary data and groups the results by match\_id and striker.
2. Rank Players by Season:
   * The result CTE calculates the total runs scored by each player for each season. It joins the player\_match\_runs with the matches and season tables to associate runs with the respective seasons. The DENSE\_RANK() function is used to rank players within each season based on their total runs.
3. Filter Top 5 Players per Season:
   * The top\_run\_rank\_player CTE filters the results to only include players who ranked between 1 and 5 in runs scored for each season and for the seasons between 2014 and 2016. The results are ordered by season\_year and season\_runs\_rank.
4. Count Player Appearances in Top 5:
   * The final query joins the top\_run\_rank\_player with the player table to get the player names. It counts how many times each player appears in the top 5 over the specified seasons, grouping the results by player ID and name.
5. Order the Results:
   * The output is ordered by the count of appearances in the top 5 in descending order.

* Summary:
* The final result will show the players who have been in the top 5 scorers list during the specified seasons, along with their names and the number of times they achieved this rank.
* Output:



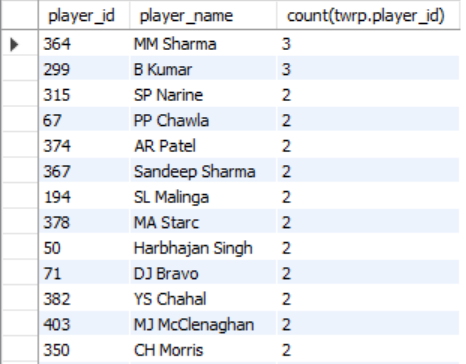
* Query: top 10 bowlers from last each 3 seasons



* Explanation:
* This SQL query identifies how many times bowlers ranked in the top 10 for wicket-takers over the last three seasons (2014 to 2016).

1. Count Wickets Taken: The player\_match\_wicket CTE calculates distinct wickets for each bowler in each match by joining the ball\_by\_ball and wicket\_taken tables.
2. Rank Bowlers by Season: The result CTE sums the total wickets taken per bowler per season and uses DENSE\_RANK() to rank them based on their performance.
3. Filter Top 10 Bowlers: The top\_wicket\_rank\_player CTE selects bowlers who ranked between 1 and 10 for total wickets in each season.
4. Count Top 10 Appearances: The final query joins this data with the player table to count how many times each bowler appeared in the top 10, grouped by player ID and name.
5. Order Results: Results are ordered by the number of appearances, showing the most frequently ranked bowlers at the top.

* Summary:
* The final result will show the bowlers who have been in the top 10 wicket-takers during the specified seasons, along with their names and the number of times they achieved this rank.
* Output:

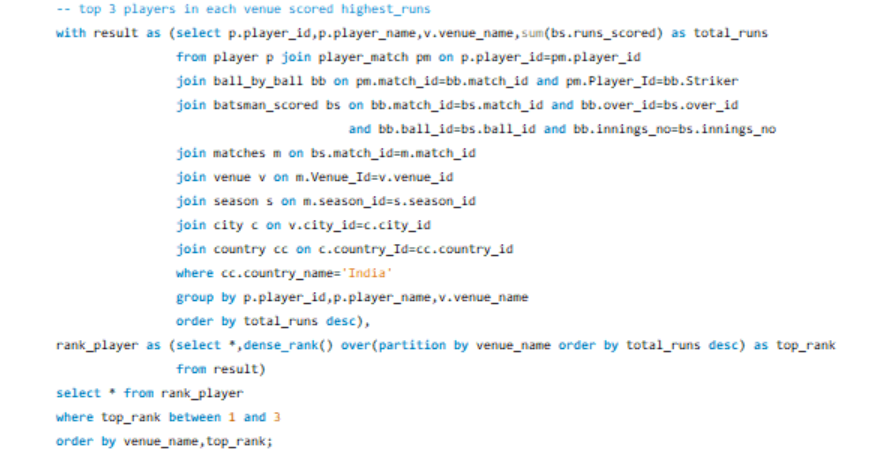


1. Are there players whose performance is more suited to specific venues or conditions? (how would you present this using charts?)

* Based on below 2 points analysed this question.

1. top 3 players in each venue scored highest runs
2. top 3 players in each venue taken highest wicket

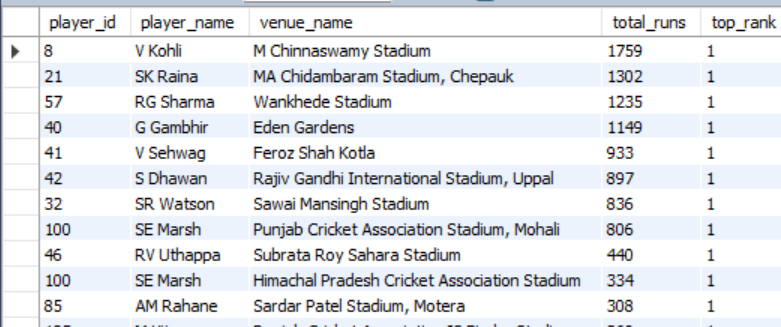
* Query: top 3 players in each venue scored highest runs.



* Explanation:
* This SQL query identifies the top three players who scored the highest runs at each venue in India. Here's a step-by-step breakdown of the process:

1. Calculate Total Runs by Player at Each Venue:
   * The result CTE aggregates the total runs scored by each player at each venue. It joins multiple tables, including player, player\_match, ball\_by\_ball, and batsman\_scored, to link players with their performances in matches played at specific venues. The query filters for matches held in India and groups the results by player and venue.
2. Rank Players by Runs Scored:
   * The rank\_player CTE ranks the players within each venue based on their total runs using the DENSE\_RANK() function. This creates a ranking for the players at each venue, allowing for ties to be handled appropriately.
3. Select Top 3 Players:
   * The final query retrieves players who rank in the top three for runs scored at each venue by filtering the results where top\_rank is between 1 and 3.
4. Order the Results:
   * The output is ordered by venue name and the ranking of the players, ensuring a clear presentation of the top scorers at each venue.

* Output:
* Original output is too big to paste here.



* Query: top 3 players in each venue taken highest wicket.

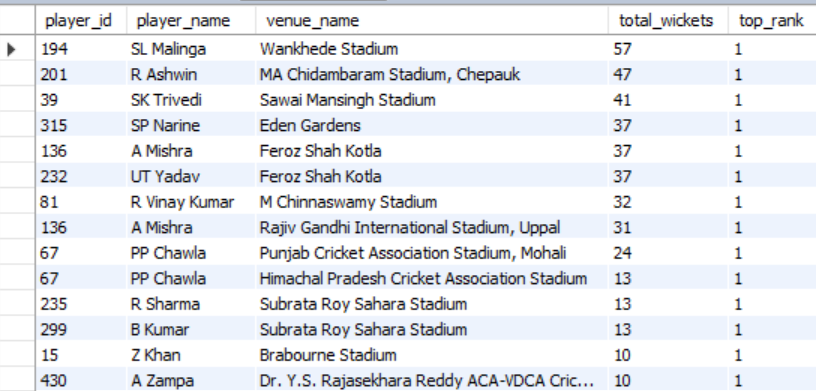


* Explanation:

This SQL query retrieves the top three wicket-takers at each cricket venue in India. Here’s a breakdown of how it works:

1. Count Wickets Taken:
   * The result Common Table Expression (CTE) counts the distinct wickets taken by each player at various venues. It joins several tables, including player, player\_match, ball\_by\_ball, wicket\_taken, matches, venue, season, city, and country.
   * The query filters for matches played in India and groups the results by player and venue, counting the distinct combinations of match ID, innings number, over ID, and ball ID to accurately reflect the number of wickets taken.
2. Ranking Players:
   * The rank\_player CTE assigns a rank to each player based on their total wickets at each venue using the DENSE\_RANK() function. This function creates a ranking within each venue based on the total wickets, allowing for ties.
3. Selecting Top 3 Wicket-Takers:
   * The final selection retrieves players ranked in the top three for each venue, ordering the results first by rank and then by total wickets taken in descending order.

* Output:



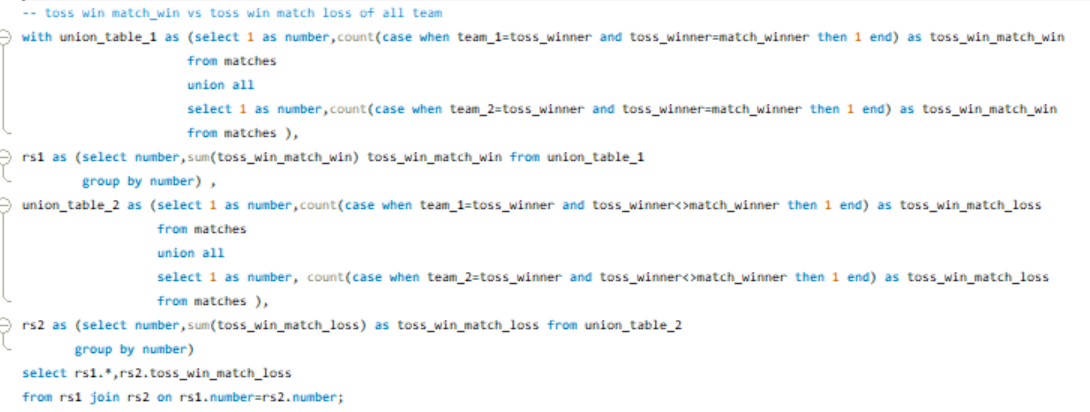
Subjective questions

1. How does toss decision have affected the result of the match? (which visualisations could be used to better present your answer) And is the impact limited to only specific venues?

* Key points

1. Toss win match win and toss win match loss of all teams.
2. Toss win match win percentage of each teams.

* Query:



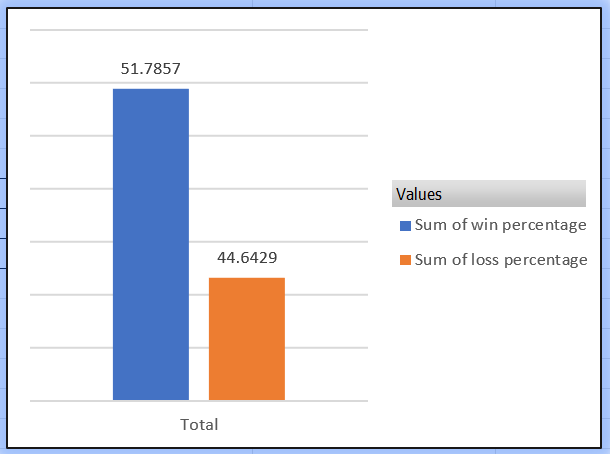
* Explanation:
* This SQL query analyzes the impact of winning the toss on match outcomes by counting two key metrics:

1. Matches Won After Winning the Toss: It counts how many times teams that won the toss also won the match, aggregating results for both team\_1 and team\_2.
2. Matches Lost After Winning the Toss: It counts how many times teams that won the toss ended up losing the match, again for both teams.

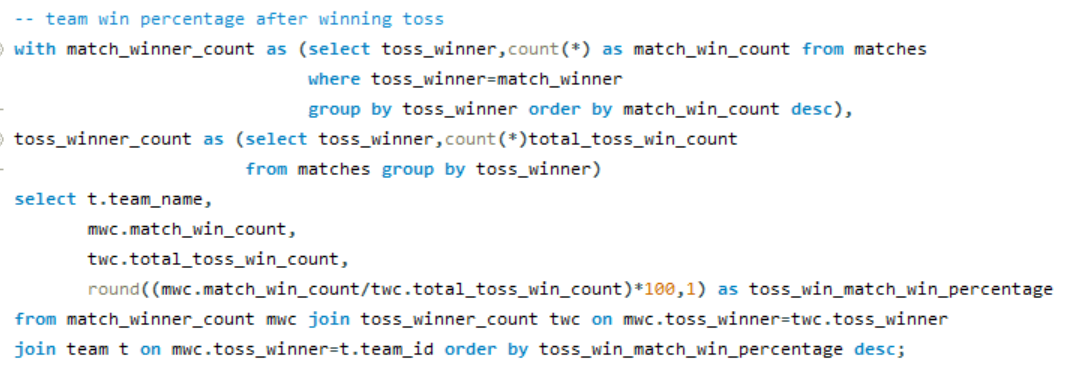
* The results from these counts are combined to show the total number of match wins and losses based on toss outcomes. This analysis helps understand whether winning the toss significantly influences match results and can be further explored by team, season, or venue.
* Outcome:



* Visualization:



* Query: Toss win and match win percentage of each team



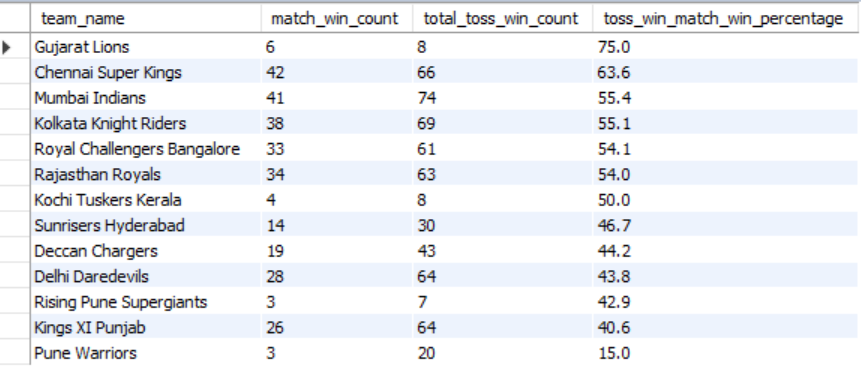
* Explanation:

This SQL query calculates the win percentage of teams after winning the toss. Here's a step-by-step explanation:

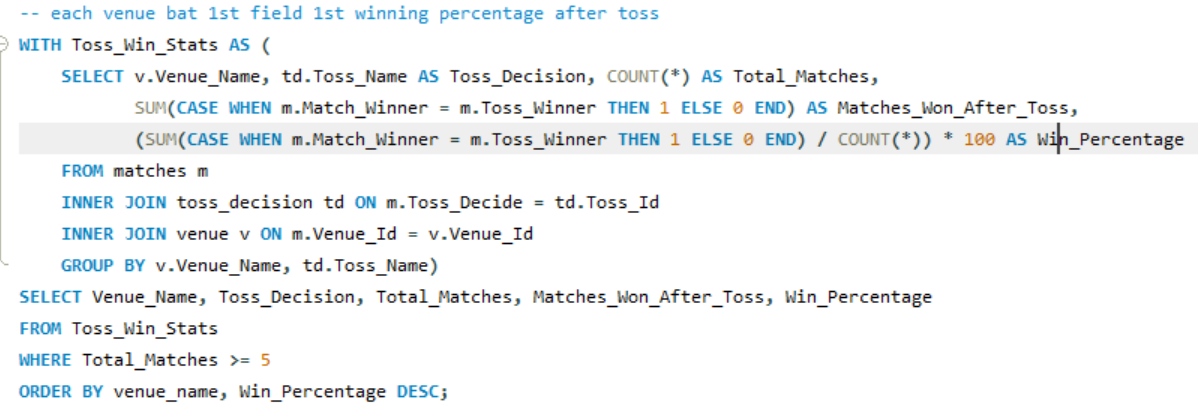
1. Count Matches Won by Toss Winners:
   * The match\_winner\_count CTE counts the number of matches won by teams that won the toss. It groups the results by toss\_winner and orders them by the count of matches won.
2. Count Total Toss Wins:
   * The toss\_winner\_count CTE counts the total number of times each team has won the toss, also grouped by toss\_winner.
3. Calculate Win Percentage:
   * The final SELECT statement joins the two CTEs (match\_winner\_count and toss\_winner\_count) to get the total number of matches won and total toss wins for each team. It then calculates the win percentage after winning the toss using the formula: Win Percentage=(Matches WonTotal Toss Wins)×100\text{Win Percentage} = \left(\frac{\text{Matches Won}}{\text{Total Toss Wins}}\right) \times 100Win Percentage=(Total Toss WinsMatches Won​)×100
   * The results are ordered by the calculated win percentage in descending order.

This query helps to identify how effectively teams capitalize on winning the toss, giving insights into whether winning the toss significantly influences their chances of winning matches.

* Output:



* Query:



* Explanation:
* This query calculates the winning percentage of teams at each venue based on their decision after winning the toss (either choosing to bat or field first).

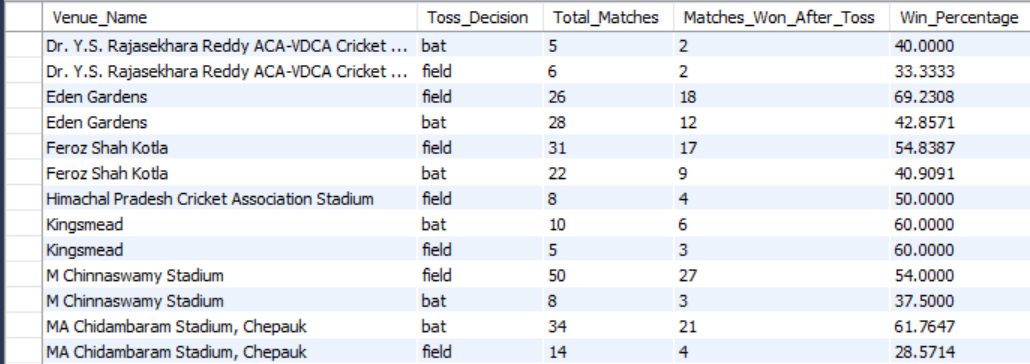
Here's a breakdown:

**Toss\_Win\_Stats CTE**: This Common Table Expression (CTE) calculates the win statistics per venue and toss decision:

* + Venue\_Name: The name of each venue.
  + Toss\_Decision: Whether the toss-winning team chose to bat or field.
  + Total\_Matches: The total number of matches at the venue where the toss was won.
  + Matches\_Won\_After\_Toss: Counts how often the toss-winning team also won the match.
  + Win\_Percentage: The percentage of matches won by the toss-winning team (calculated by dividing Matches\_Won\_After\_Toss by Total\_Matches and multiplying by 100).

**Final Query**:

* + Retrieves each venue's toss statistics for cases where at least 5 matches have been played (WHERE Total\_Matches >= 5).
  + Orders the result by venue\_name alphabetically and Win\_Percentage in descending order to see the highest winning percentages first.
* Output:



* Insights:
* Choosing right decision will have the crucial impact on team win/loss.
* As per the analysis specific venues is most suitable for chasing and some venues most suitable for batting 1st and some venues is 50 50 ration it depends on team and weather.
* \*\*\*Example: - M Chinna swamy stadium (RCB Home ground) is most suitable for chasing

And team with chasing is having high winning percentage in this ground compare to batting 1st team.

1. Suggest some of the players who would be best fit for the team?

* Considering the below points suggested the best players for the team

1. \* Top 5 score rank in each season--players and number of times they are in top 5 in last 3 seasons.
2. \* Top 10 player with highest strike rate and minimum 1500 runs scored.
3. \* Top 10 bowlers taken highest wicket.
4. \* Top 10 bowlers with best economy, minimum 100 overs bowled.

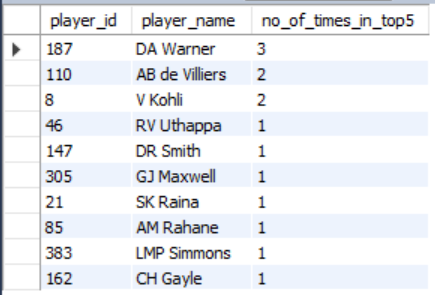
* Query:



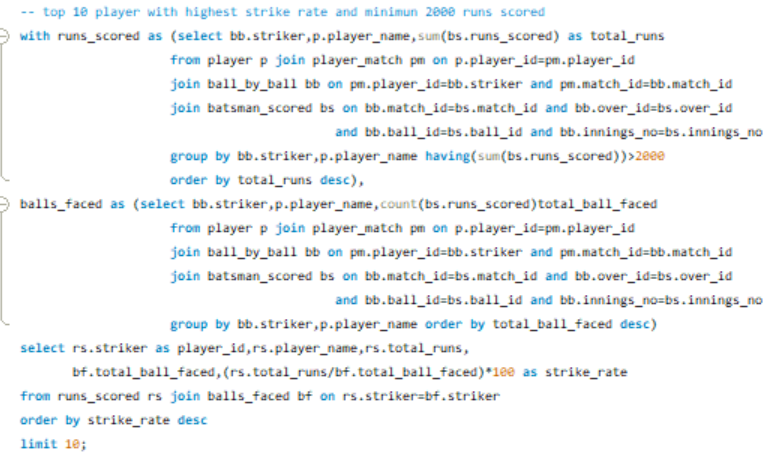
* Explanation:
* This query identifies players who ranked in the top 5 scorers for each season from 2014 to 2016 and counts how often each player achieved this ranking. It returns the top 10 players with the highest counts.

**Steps:**

* **Calculate Match Runs**: Get total runs scored by each player in each match.
* **Season Rankings**: Rank players by total season runs for each season, assigning a rank within each season.
* **Filter Top 5 Players**: Select players who ranked in the top 5 from 2014 to 2016.
* **Count Top 5 Appearances**: Count how many times each player was in the top 5 over these seasons and return the top 10 players by frequency.
* Output:



* Query:

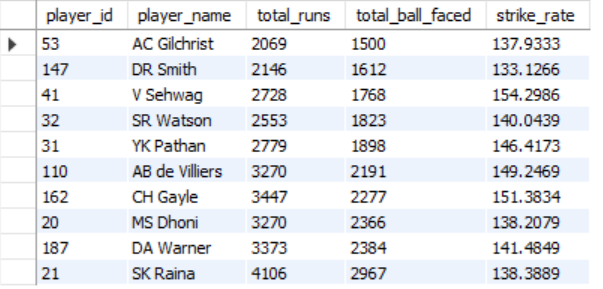


Explanation:

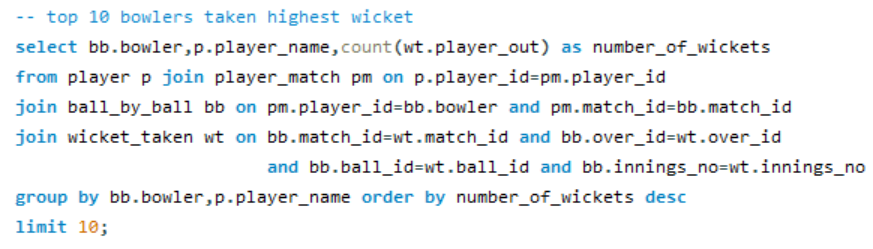
* This query identifies the top 10 players with the highest strike rate who have scored at least 2000 runs. Here’s a brief explanation of each step:
* **runs\_scored CTE**:
  + Calculates the total runs scored by each player.
  + Only includes players with more than 2000 total runs (HAVING (SUM(bs.runs\_scored)) > 2000).
  + Orders players by total\_runs in descending order.
* **balls\_faced CTE**:
  + Counts the total balls faced by each player (each runs\_scored entry).
  + Groups by player and orders by total\_ball\_faced in descending order.
* **Final Query**:
  + Joins runs\_scored and balls\_faced on player ID to obtain both total runs and balls faced.
  + Calculates each player’s strike rate as (total\_runs / total\_ball\_faced) \* 100.
  + Orders players by strike rate in descending order and limits the result to the top 10.

This final list shows the top 10 players with the highest strike rates among those who have scored over 2000 runs.

* Output:



* Query:



* Explanation:
* This query retrieves the top 10 bowlers who have taken the highest number of wickets. Here's a brief breakdown:
* **Join Tables**:
  + Joins the player, player\_match, ball\_by\_ball, and wicket\_taken tables to gather data on bowlers and their wickets in each match.
* **Calculate Wickets**:

* + Counts the occurrences of player\_out in the wicket\_taken table for each bowler, representing the number of wickets taken by that bowler.
* **Group and Order**:
  + Groups results by bowler and player\_name to get the total wickets for each bowler.
  + Orders by number\_of\_wickets in descending order and limits the result to the top 10 bowlers.
* This final output shows the 10 bowlers with the most wickets, ranked by the highest wicket count
* Output:



* Query:



* Explanation:

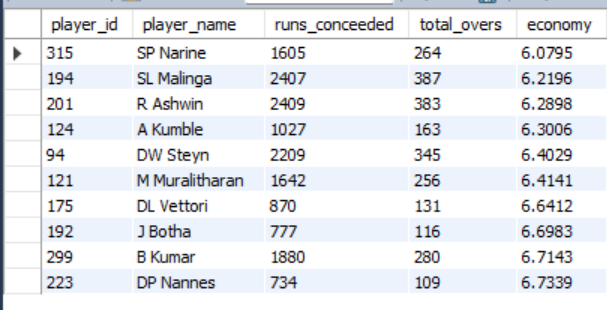
This query identifies the top 10 bowlers with the best economy rates (lowest runs conceded per over) who have bowled at least 100 overs.

**Steps:**

* **Calculate Runs Conceded**: Sum up total runs conceded by each bowler.
* **Calculate Overs Bowled**: Count unique overs bowled by each bowler.
* **Compute Economy Rate**: Divide runs conceded by overs bowled for each bowler with more than 100 overs.
* **Filter and Sort**: Select the top 10 bowlers with the lowest economy rates.

This provides the 10 most economical bowlers with a minimum threshold of overs bowled.

* Output:



Suggestion:

* Warner, Kohli, AB De Villiers, Raina, Gayle are the top players with most consistently in the top 10 list of last 3 seasons.
* Warner, AB De Villiers, Raina, Gayle, , Watson are also having the best strike rate

* Mallinga, Bravo, Mishra, Harbajan Singh are the top bowlers with highest wicket taken.
* Narain, Malinga, Kumble, Ashwin, Steyn are the bowlers with best economy. considering lowest runs per over.
* By considering this above player are the best suitable for RCB.

1. What are some of parameters that should be focused while selecting the players?

* When selecting players for a cricket team several key parameters should be considered to ensure a well-rounded and competitive squad. Here are some important factors:

1. **Performance Statistics**:
   * **Batting Average**: Indicates consistency and reliability.
   * **Strike Rate**: Measures how quickly a player scores, important for limited-overs formats.
   * **Bowling Average**: Reflects effectiveness in taking wickets.
   * **Economy Rate**: Shows how many runs a bowler concedes per over, crucial for assessing bowlers in limited-overs games.
   * **Number of Wickets**: For bowlers, indicating their ability to take key wickets.
2. **Fitness and Physical Condition**:
   * **Injury History**: Players with a history of injuries may be a risk.
   * **Fitness Levels**: Overall fitness impacts performance and endurance.
3. **Skill Set**:
   * **Fielding Ability**: A player’s agility and catching skills can make a significant difference.
   * **Specialization**: Identifying players who excel in specific roles (e.g., opening batsman, all-rounder, wicketkeeper) to balance the team.
4. **Consistent Performance**:
   * **Consistency**: Players who regularly perform well are often more valuable.
5. **Experience**:
   * **Match Experience**: Experience in international or high-pressure games can be crucial.
   * **Leadership Qualities**: Players with leadership experience can guide younger teammates.
6. **Adaptability**:
   * **Ability to Play in Different Conditions**: Players should be able to adjust to various pitches and weather conditions.
   * **Versatility**: Players who can perform well in multiple roles (e.g., batting at different positions) can provide strategic flexibility.
7. **Mental Toughness**:
   * **Pressure Handling**: Ability to perform under pressure is vital, especially in critical matches.
   * **Attitude and Work Ethic**: A positive mindset and dedication can enhance team dynamics.
8. Which players offer versatility in their skills and can contribute effectively with both bat and ball? (can you visualize the data for the same).

* Query:



Explanation:

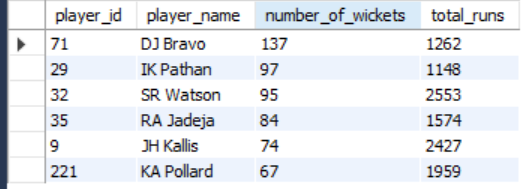
* This SQL query identifies the top all-rounders in a cricket database who have scored more than 1000 runs and taken more than 50 wickets. Here's a breakdown of how it works:

**Steps:**

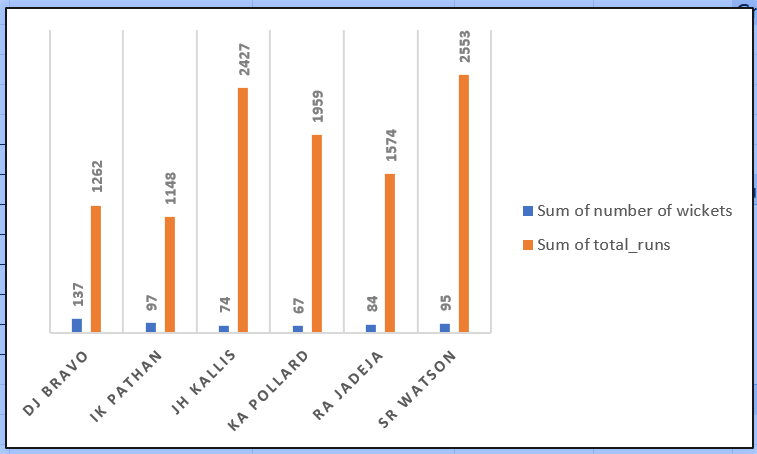
* **Identify Top Bowlers**:
  + The **top\_bowler** Common Table Expression (CTE) aggregates the total wickets taken by each bowler.
  + It joins relevant tables to count the number of players dismissed (wickets taken) for each bowler based on their performance in matches.
* **Identify Top Batsmen**:
  + The **top\_batsman** CTE sums the total runs scored by each batsman.
  + It also joins the necessary tables to calculate the total runs based on individual performances in matches.
* **Combine Results**:
  + The main query selects player details from both CTEs, where it matches bowlers who are also batsmen (indicating they are all-rounders).
  + The selection criteria filter for players who have taken more than 50 wickets and scored more than 1000 runs.

**Result:**

* The output includes the player ID, player name, the number of wickets taken, and the total runs scored for each qualifying all-rounder.
* This approach effectively filters and combines performance metrics from both batting and bowling disciplines to identify all-rounders in the dataset.
* Output:



* Visualization:



* Insights:
* Player with batting performance and bowling including are well required in critical situation. They can adapt to both the situation.in most of the times match winning situation these players contribute lot to the team.

1. Are there players whose presence positively influences the morale and performance of the team? (justify your answer using visualisation).

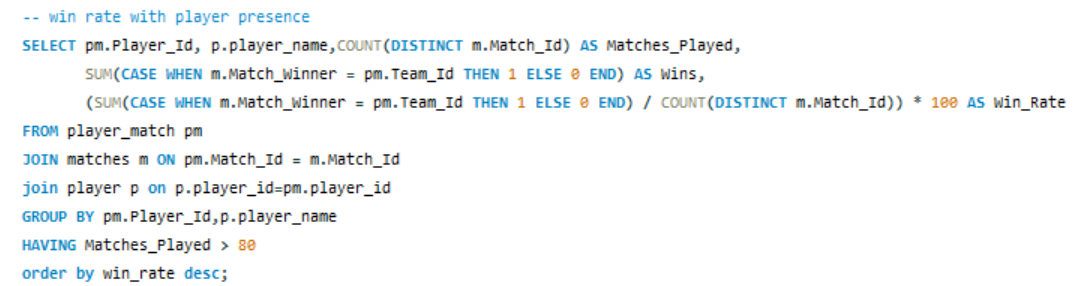
* To determine if there are players whose presence positively influences the morale and performance of the team, we can analyse various metrics related to the players, their roles, and their contributions to the team's success. Here are some approaches.

**Win Rates with Player Participation**:

* Analyse the win rates of teams when certain players are part of the lineup compared to when they are not. Higher win rates with their participation might indicate a positive influence.

**Man of the Match Awards**:

* Evaluate how often players receive "Man of the Match" awards, as these reflect significant individual contributions to match outcomes.
* Query:



* Explanation:

This SQL query calculates the win rate of players based on their presence in matches. Here’s a short breakdown of its components and functionality:

Explanation of the Query

FROM and JOIN Clauses:

* + The query starts from the player\_match table (pm), which connects players to specific matches.
  + It joins the matches table (m) to access match-related information, such as the match winner.
  + It also joins the player table (p) to get the player's name.

Selecting Fields:

* + pm.Player\_Id and p.player\_name: These columns identify each player.
  + COUNT(DISTINCT m.Match\_Id) AS Matches\_Played: Counts the total number of unique matches each player has participated in.
  + SUM(CASE WHEN m.Match\_Winner = pm.Team\_Id THEN 1 ELSE 0 END) AS Wins: Sums up the number of matches won by the player's team. If the player's team won, it counts as a win; otherwise, it counts as zero.

Calculating Win Rate:

* + The win rate is calculated as the total wins divided by the total matches played, multiplied by 100 to convert it to a percentage.

GROUP BY Clause:

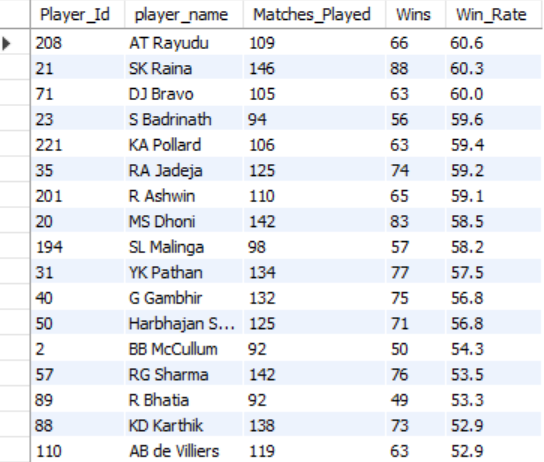
* + Groups the results by player ID and name, allowing aggregation functions (like COUNT and SUM) to operate within each group.

HAVING Clause:

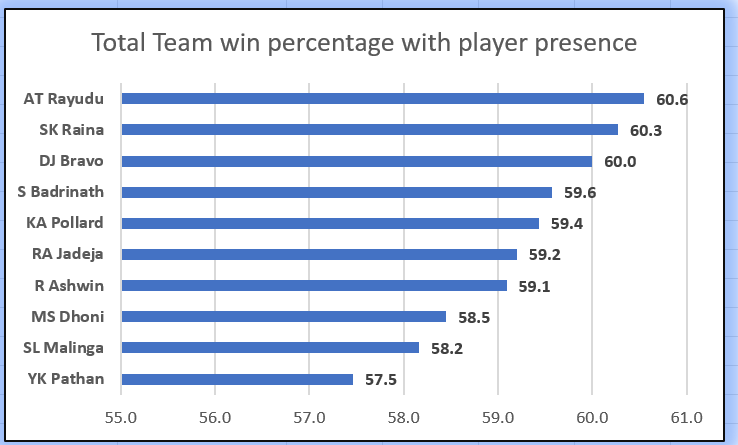
* + Filters the results to only include players who have participated in more than 80 matches.

ORDER BY Clause:

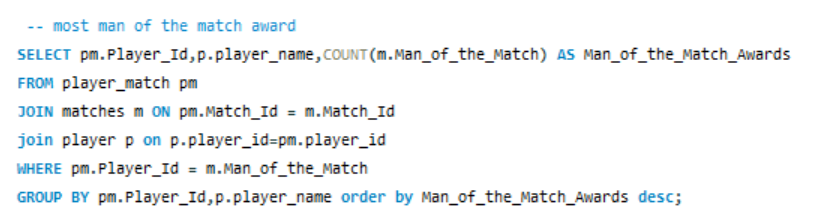
* + Orders the final results by win rate in descending order, so the players with the highest win rates appear first.
* Output:



* Visualization:



* Query:

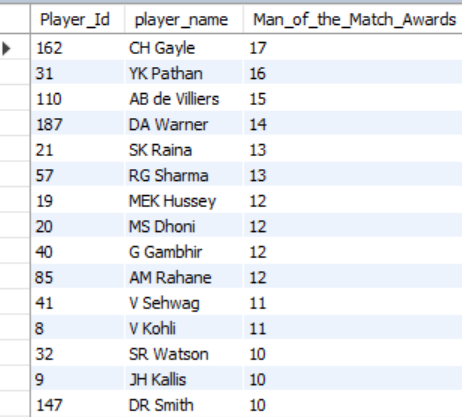


* Explanation:

This SQL query retrieves the players who have won the most "Man of the Match" awards in cricket. It does this by:

1. **Joining Tables**: It combines data from the player\_match, matches, and player tables to correlate players with their match performance.
2. **Counting Awards**: It counts the number of times each player has been listed as the "Man of the Match" by comparing player IDs with the Man\_of\_the\_Match field in the matches table.
3. **Grouping Results**: It groups the results by player ID and name to aggregate the award counts.
4. **Ordering Results**: Finally, it orders the output in descending order, showcasing the players with the highest number of "Man of the Match" awards at the top.

* Output:



* Insights:
* By considering above two factors AB de Villiers, Raina, MS Dhoni, Pathan, RG Sharma

Are the players with both most man of the match award recievers and in the team with winning percentage greater than 50.

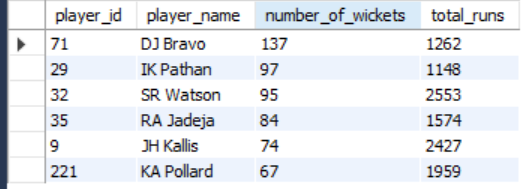
* By combining these metrics, you can identify players whose presence may positively influence the team's morale and performance. The analyses can reveal patterns that indicate the impact of individual players on the overall success of the team.

1. What would you suggest to RCB before going to mega auction?

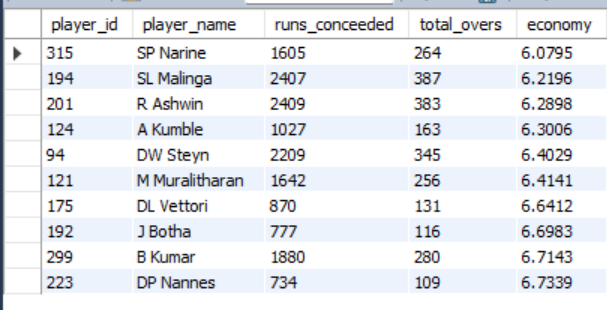
* Before the mega auction, Ill suggest Royal Challengers Bangalore (RCB) to consider the following strategies and recommendations.
* \* Players with highest runs.
* \* Players with best average.
* \* Players with most strike rate.
* \* Players with most wickets taken.
* \* Players with best economy in bowling.
* \* Players with most success rate in team winning including highest man of the match award.etc

By considering the above factors the below players ill suggest to RCB before going to mega auction.

* All-rounders:



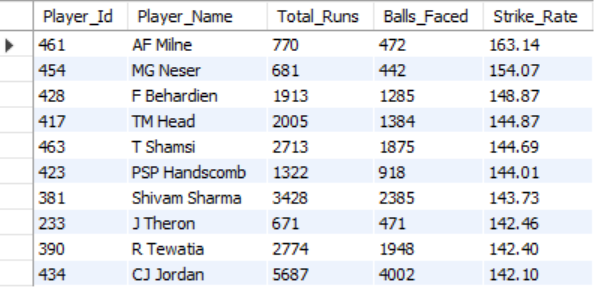
* Bowler with best economy:



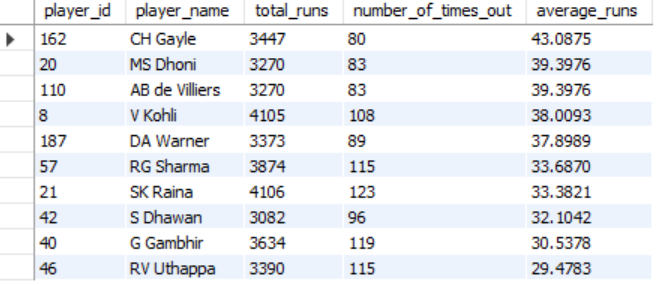
* Bowlers with best wicket taking ability:



* Players with top run rate:



* Players with best average - minimum (300 runs scored)



* Above players are the best players and ill suggest RCB before mega Auction.

1. What do you think could be the factors contributing to the high-scoring matches and the impact on viewership and team strategies.

High-scoring matches in cricket can be attributed to various factors, each influencing viewership and team strategies in significant ways. Here are some key contributors:

**Factors Contributing to High-Scoring Matches**

* **Pitch Conditions**:
  + **Batting-Friendly Pitches**: Flat surfaces with good bounce and minimal seam movement allow batsmen to score freely.
  + **Dew Factor**: In day-night matches, the presence of dew can make it harder for bowlers to grip the ball, leading to more runs.
* **Format of the Game**:
  + **Limited Overs**: T20 and One-Day Internationals (ODIs) are designed for quick scoring, with rules that favor aggressive batting.
  + **Powerplay Rules**: Restrictions on field placements during powerplays encourage aggressive batting, resulting in higher scores.
* **Batting Powerhouses**:
  + **Strong Batting Lineups**: Teams with several explosive batsmen can chase or set high targets more easily.
  + **In-Form Players**: Individual performances from star players can significantly inflate scores.
* **Bowling Quality**:
  + **Weak Bowling Attacks**: If a team has inexperienced or underperforming bowlers, it can lead to higher scores from the opposing team.
  + **Injury to Key Bowlers**: Missing out on key bowlers can weaken the team's ability to control runs.
* **Game Strategy**:
  + **Aggressive Intent**: Teams adopting an aggressive approach from the outset can result in high scoring.
  + **Chasing Teams**: Teams that are comfortable chasing targets tend to take more risks, contributing to higher scores.
* **Fielding Errors**:
  + **Dropped Catches and Misfields**: Mistakes in the field can allow batsmen to score runs more easily, raising overall match scores.
* **Use of Technology**:
  + **Data Analytics**: Teams increasingly use analytics to exploit weaknesses in opposition bowling and fielding, leading to more aggressive strategies.

**Impact on Viewership**

1. **Entertainment Value**:
   * High-scoring matches are typically more exciting and can attract larger audiences due to the fast-paced action and frequent boundary scoring.
2. **Increased Engagement**:
   * Fans enjoy seeing big scores and thrilling moments, which can enhance viewer engagement both during the match and on social media.
3. **Marketing and Sponsorship**:
   * High-scoring matches often lead to better visibility for sponsors and higher advertising revenues, making them attractive to broadcasters.
4. **Increased Ratings**:
   * Matches that feature high scores are likely to achieve better television ratings and higher attendance at stadiums.

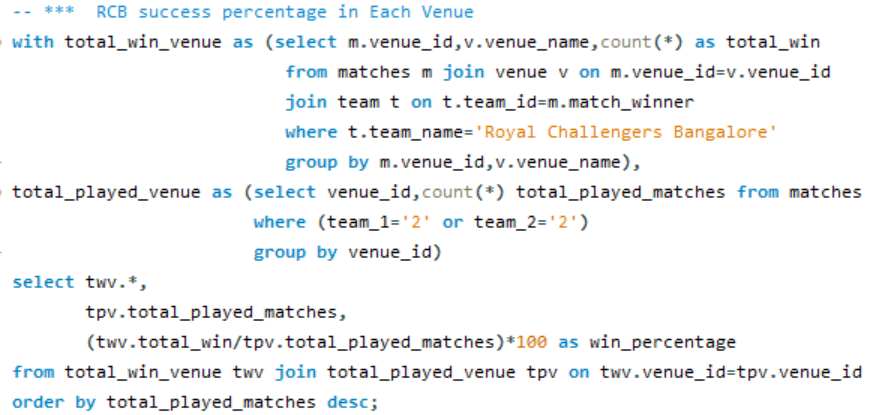
**Impact on Team Strategies**

1. **Adjusting Game Plans**:
   * Teams may adopt more aggressive batting strategies and focus on power hitters in their lineup if high scores are anticipated.
2. **Bowling Tactics**:
   * Teams may emphasize bowling variations and tactics, such as slower balls and yorkers, to counteract aggressive batting.
3. **Player Selection**:
   * Teams might prioritize players with a proven ability to score quickly or bowl economically in high-pressure scenarios.
4. **Field Placements**:
   * Captains may change their field placements to encourage wickets rather than trying to contain runs, leading to a more attacking mindset.
5. **Preparation and Training**:
   * Teams may focus on enhancing their fitness levels and skill sets to cope with high-pressure, high-scoring games.

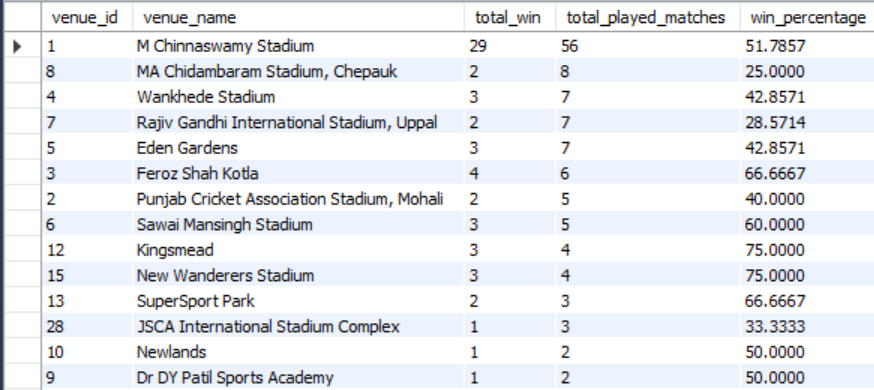
In summary, high-scoring matches create a dynamic interplay of entertainment, strategic planning, and fan engagement, influencing not only how teams approach their game but also how cricket is marketed and consumed by audiences.

1. Analyse the impact of home ground advantage on team performance and identify strategies to maximize this advantage for RCB.

* Key points on home ground.
* The concept of home ground advantage in sports has a significant impact on team performance. Analysing this effect involves understanding how familiar conditions, fan support, and the psychological comfort of playing at home contribute to a team’s success. Here’s a breakdown of the impact and key factors involved in maximizing this advantage:
* **Pitch and Weather Conditions**: Teams are familiar with their home ground’s pitch behaviour, such as whether it Favors batting, spin, or pace.
* **Crowd Influence**: Home crowd support boosts player morale and can pressure the visiting team.
* **Win Rate**: Teams often have higher win rates at home compared to away matches. By analysing historical win rates, teams can evaluate the extent of their advantage when playing on their home turf.
* **Batting and Bowling Averages**: Batsmen and bowlers often perform differently at home versus away due to familiarity with pitch conditions. Analysing these averages can reveal whether players consistently perform better at home.
* **Fielding Metrics:** Teams tend to make fewer fielding errors at home due to familiarity with ground dimensions, sightlines, and lighting, which can reduce the likelihood of dropped catches or mis fielding incidents.
* Case Study: Royal Challengers Bangalore (RCB) and Home Ground Impact.
* Query:



* Output and Insight:



* Total played matches for every team are more in home ground compare to other venue.

It will increase the chance of winning the team.

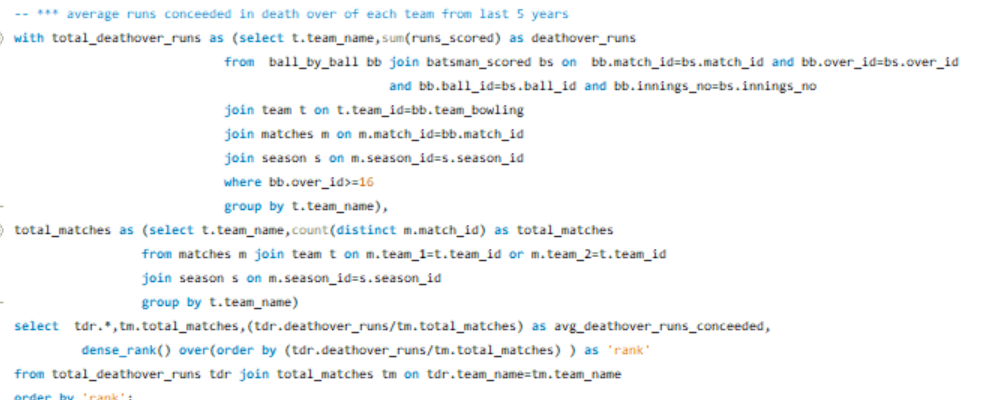
* RCB have not great winning percentage in home ground with 51.78%.
* **High-Scoring Games**: The stadium’s batting-friendly conditions lead to high scores, often requiring RCB to maintain a powerful batting lineup.
* **Conclusion**
* The home ground advantage in cricket has a clear impact on team performance, often providing a boost through familiarity, fan support, and reduced fatigue. By strategically selecting condition-specific players, enhancing fan engagement, and preparing through data-driven analysis, teams can maximize their home ground advantage. However, they must manage expectations and avoid complacency to ensure consistent performance.

1. Come up with a visual and analytical analysis with the RCB past seasons performance and potential reasons for them not winning a trophy.

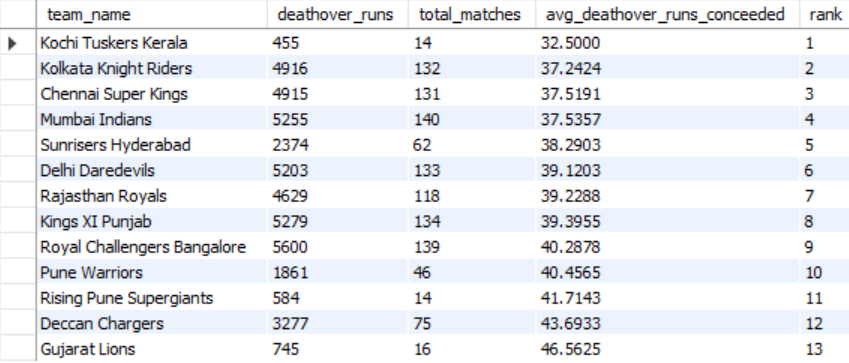
* Possible reason:
* RCB is having the very strong batting line up in past seasons. In Top 5 scorers and top 5 average score players from all past season,3 players from RCB alone (players are Virat Kohli, AB de Villiers, Gayle).
* RCB death over performance in batting is strong compare to other teams.
* This make RCB very strong in batting line up.
* **Over-Reliance on Key Players**  
  RCB has often relied heavily on star players like Virat Kohli, AB de Villiers, Gayle. This over-dependence has sometimes led to issues when these stars failed to perform in crucial matches, leaving little backup from the rest of the team.
* Inconsistent bowling performance is the reason to not achieve the trophy.
* **Home ground performance:**

RCB is not having the very good home ground performance compare to other strong team like CSK and MI. (RCB home ground success percentage 51.8) this is also the main reason for RCB to struggle to get trophy.

* **Death over bowling performance:**
* **Query:**

****

* **Output:**

****

* RCB’s death overs runs conceded average is 40.28 is very high compare to the top rank teams.
* Conclusion:
* Overdependency on key players.
* Inconsistence bowling line up.
* bad home ground performance.

These are the potential reason for RCB to not win the trophy

1. How would you approach this problem, if the objective and subjective questions weren't given?

If objective and subjective questions weren’t provided, here’s how I’d approach analyzing RCB’s performance and identifying areas for improvement:

**Step 1: Define Key Performance Indicators (KPIs)**

Since we don’t have predefined questions, I would first identify the most meaningful KPIs for a team’s performance. For RCB, we might look at:

1. **Batting Metrics**
   * **Total Runs Scored per Season**: Helps understand team scoring capability over time.
   * **Strike Rate of Key Players**: Measures efficiency and ability to capitalize on deliveries.
   * **Consistency of Run Scorers**: Evaluates top batsmen’s reliability across matches and seasons.
   * **Boundary Count (4s and 6s)**: Indicates ability to score quickly, especially relevant for T20s.
2. **Bowling Metrics**
   * **Wickets Taken per Season**: Measures the effectiveness of the bowling unit.
   * **Bowling Economy**: Tracks efficiency in terms of runs allowed per over.
   * **Dot Ball Percentage**: Analyzes pressure applied by bowlers.
   * **Strike Rate of Bowlers**: Shows average deliveries per wicket, essential for evaluating impact.
3. **Fielding Metrics**
   * **Catches, Run-Outs, and Stumpings**: Measures fielding impact, which is crucial in close games.
   * **Fielding Efficiency**: Assesses contribution in reducing the opposition’s scoring.
4. **Team Metrics**
   * **Win/Loss Ratio per Season**: Indicates overall success rate.
   * **Home vs. Away Performance**: Identifies any home ground advantage.
   * **Performance Trends Over Seasons**: Tracks whether the team is improving or declining.

**Step 2: Analyze and Aggregate Data**

With KPIs identified, I’d retrieve relevant data from the database tables and create aggregated views. Key steps here would include:

* **Season-wise Aggregations**: For each KPI, aggregate season-wise data for runs, strike rates, economy rates, etc., by players and by team to create a time series of RCB’s performance.

**Step 3: Derive Insights from Data**

Once the data is structured and aggregated, I’d analyse it for patterns:

1. **Identify Strong Performers**:
2. **Explore Batting and Bowling Patterns**:
3. **Season-by-Season Comparisons**:
4. **Situational Insights**:
   * **Toss Decisions**: Assess how toss decisions impact outcomes, especially at different venues.
   * **Impact of Partnerships**: Analyse batting partnerships to identify productive pairs.
   * **Role of Key Players**: Examine the influence of star players on match outcomes by looking at matches with and without these players.

**Step 4: Develop Strategic Recommendations**

Based on these analyses, I’d form strategic recommendations:

* **Squad Composition**
* **Venue-Based Strategies**: Create specific strategies for home and away games based on historical performance. For example, if RCB performs better when defending totals at M. Chinna swamy Stadium, they could prioritize bowlers who can handle high-pressure chases.

**Step 5: Visualization and Dashboard Creation**

To make insights accessible, I’d design visualizations:

1. **Season Performance Trends**: Line charts for runs scored, wickets taken, and win/loss ratios over the seasons.
2. **Player Heatmaps**: Show player performances across different venues and conditions.
3. **Bowling vs. Batting Impact**: Dual-axis charts comparing team batting and bowling metrics season by season.
4. **Contribution Analysis**: Pie or bar charts to show player contributions to key metrics, such as total runs and wickets.
5. **Home and Away Advantage**: Comparative bar charts showing win/loss ratios at different venues.
6. In the "Match" table, some entries in the "Opponent\_Team" column are incorrectly spelled as "Delhi\_Capitals" instead of "Delhi\_Daredevils". Write an SQL query to replace all occurrences of "Delhi\_Capitals" with "Delhi\_Daredevils".

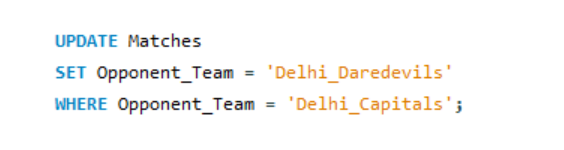
* There is no Match table exist and no column with Opponent\_team in any of the table.

The available column with team name is Team\_name in table Team.

And there is no such name available like "Delhi\_Capitals"

If the table and column exist the query will be like this

* Query:



* **Explanation**
* **UPDATE Match**: Targets the Match table for updates.
* **SET Opponent\_Team = 'Delhi\_Daredevils'**: Replaces the value in the Opponent\_Team column with "Delhi\_Daredevils" for specific rows.
* **WHERE Opponent\_Team = 'Delhi\_Capitals'**: Ensures that only rows with the value "Delhi\_Capitals" in Opponent\_Team are updated, preventing unintended changes.

After running this query, all instances of "Delhi\_Capitals" in Opponent\_Team will be corrected to "Delhi\_Daredevils.