**Tasks**

**Objective Questions & Answers**

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

USE ipl;

SELECT COLUMN\_NAME, DATA\_TYPE

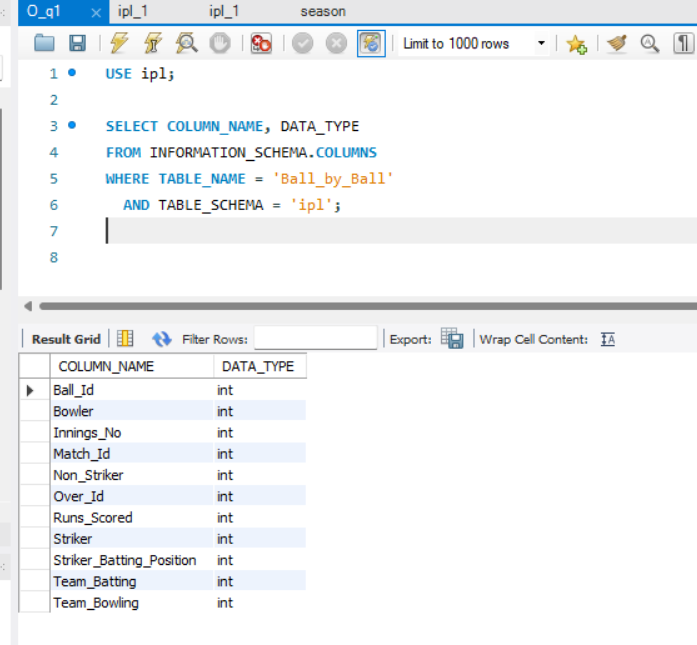
FROM INFORMATION\_SCHEMA.COLUMNS

WHERE TABLE\_NAME = 'Ball\_by\_Ball'

AND TABLE\_SCHEMA = 'ipl';

**Explanation:**

* **What is INFORMATION\_SCHEMA.COLUMNS?**
* It’s a **system table** that stores **metadata** about every column in every table in every database.
* Use it to find column names, data types, nullability, etc.
* **What this query does:**
* COLUMN\_NAME: Returns the names of all columns in the Ball\_by\_Ball table.
* DATA\_TYPE: Returns the type of data stored in each column (like int, varchar, datetime).
* TABLE\_SCHEMA = 'ipl': Ensures we are looking specifically inside your IPL database.
* **Expected Output:**



* **Why this matters:**

Understanding data types helps when:

* Writing queries (e.g., no quotes for integers)
* Joining tables correctly (matching IDs)
* Avoiding data type mismatch errors

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

**Step 1: Confirm RCB’s Team ID**

SELECT Team\_Id, Team\_Name

FROM Team

WHERE Team\_Name LIKE '%Royal Challengers%';

Result:

* it returns Team\_Id = 2

**Step 2: Find all matches in Season 1 where RCB was involved**

SELECT Match\_Id, Team\_1, Team\_2

FROM Matches

WHERE Season\_Id = 1

AND (Team\_1 = 2 OR Team\_2 = 2);

**Result:**

* **No rows returned**, which means **RCB did not play any matches in Season 1**
* **Conclusion:** RCB **did not participate** in Season 1

**Step 3: Check which teams actually played in Season 1**

SELECT DISTINCT Team\_1 AS Team\_Id FROM Matches WHERE Season\_Id = 1

UNION

SELECT DISTINCT Team\_2 AS Team\_Id FROM Matches WHERE Season\_Id = 1;

**Step 4: Verify ball-by-ball data also doesn't exist for Season 1**

SELECT DISTINCT m.Season\_Id

FROM Matches m

JOIN Ball\_by\_Ball bb ON m.Match\_Id = bb.Match\_Id;

**Result:**

* Only Season 6, 7, 8, 9 are present
* Season 1 data is missing from Ball\_by\_Ball
* RCB (Royal Challengers Bangalore) did **not participate in Season 1**, based on the Matches table — there are no match records where RCB’s Team\_Id appears.
* Additionally, the Ball\_by\_Ball table contains **no records for Season 1**, so even if they had played, we would not be able to calculate their total runs.

3.How many players were more than the age of 25 during season 2014?

**Step 1: Understand what’s needed**

We need to:

* Get players from the player table.
* Calculate their age at the start of **IPL Season 2014**.
* Count how many of them were **older than 25**.

**Step 2: Identify useful tables and columns**

From schema:

* Table: player
  + Column: DOB → Player’s date of birth
* Table: season
  + Column: Season\_Year → Year of IPL season (but no actual date column)

Since there is **no exact start date**, we’ll assume IPL started on **1st April 2014**.

**Step 3: Use SQL to calculate age**

* We use TIMESTAMPDIFF(YEAR, DOB, '2014-04-01') to calculate a player’s age as of April 1, 2014.
* TIMESTAMPDIFF(YEAR, DOB, '2014-04-01') = Age of player in years

**Step 4: Filter players older than 25**

* Now we filter only those where age is greater than 25.

WHERE TIMESTAMPDIFF(YEAR, DOB, '2014-04-01') > 25

**Step 5: Count how many such players**

* We use COUNT(\*) to get the total number of such players.

SELECT COUNT(\*) AS Players\_Above\_25

FROM player

WHERE TIMESTAMPDIFF(YEAR, DOB, '2014-04-01') > 25;

**What this does:**

* Checks each player's age as of April 1, 2014
* Filters players whose age > 25
* Counts them

Plyers above 25 = 324

4.How many matches did RCB win in 2013?

**Step 1: Understand the tables needed**

From the schema, these tables are relevant:

* team: has Team\_Id, Team\_Name
* season: has Season\_Id, Season\_Year
* matches: has:
  + Match\_Id
  + Season\_Id → connects to season
  + Match\_Winner → contains the Team\_Id of the match winner

**Step 2: Get RCB’s Team\_Id**

Run this to get the Team\_Id of **RCB**:

SELECT Team\_Id

FROM team

WHERE Team\_Name = 'Royal Challengers Bangalore';

Let's say it returns 2. We'll use this in the next query.

**Step 3: Get the Season\_Id for 2013**

Run this:

SELECT Season\_Id

FROM season

WHERE Season\_Year = 2013;

this returns 6. We'll use this too.

**Step 4: Count matches RCB won in 2013**

Now combine both IDs and count matches from the matches table:

SELECT COUNT(\*) AS RCB\_Wins\_2013

FROM matches

WHERE Match\_Winner = 2 -- RCB's Team\_Id

AND Season\_Id = 6; -- 2013 season

**Explanation:**

* Match\_Winner = 2: Only matches where RCB was the winner.
* Season\_Id = 6: Only matches played in the 2013 season.
* COUNT(\*): Count how many such matches exist.

**Final Output:**

This query will return the **number of matches RCB won in 2013**.

**RCB win 9 matchs in 2013.**

**5.List the top 10 players according to their strike rate in the last 4 seasons**

**Step 1: Understand Strike Rate**

The strike rate in cricket is calculated as:

**Strike Rate = (Total Runs Scored / Total Balls Faced) \* 100**

**Step 2: Find the last 4 seasons**

SELECT Season\_Id, Season\_Year

FROM season

ORDER BY Season\_Year DESC

LIMIT 4;

this returns Season\_Id: 9, 8, 7, 6 (latest 4 years).

**Step 3: Use these Season\_IDs to filter matches**

We'll get all Match\_Ids for these seasons:

sql

CopyEdit

SELECT Match\_Id

FROM matches

WHERE Season\_Id IN (6, 7, 8, 9);

**Step 4: Join with ball\_by\_ball to get Runs and Balls Faced**

We'll now calculate:

* SUM(Runs\_Scored) → Total runs by a player
* COUNT(\*) → Total balls faced (each row = one ball)

**Step 5: Write Final Query**

SELECT

bb.Striker as Player\_Id,

p.Player\_Name,

SUM(bb.Runs\_Scored) AS Total\_Runs,

COUNT(\*) AS Balls\_Faced,

ROUND((SUM(bb.Runs\_Scored) / COUNT(\*)) \* 100, 2) AS Strike\_Rate

FROM

ball\_by\_ball bb

JOIN

matches m ON bb.Match\_Id = m.Match\_Id

JOIN

player p ON bb.Striker = p.Player\_Id

WHERE

m.Season\_Id IN (6, 7, 8, 9) -- last 4 seasons

GROUP BY

bb.Striker

HAVING

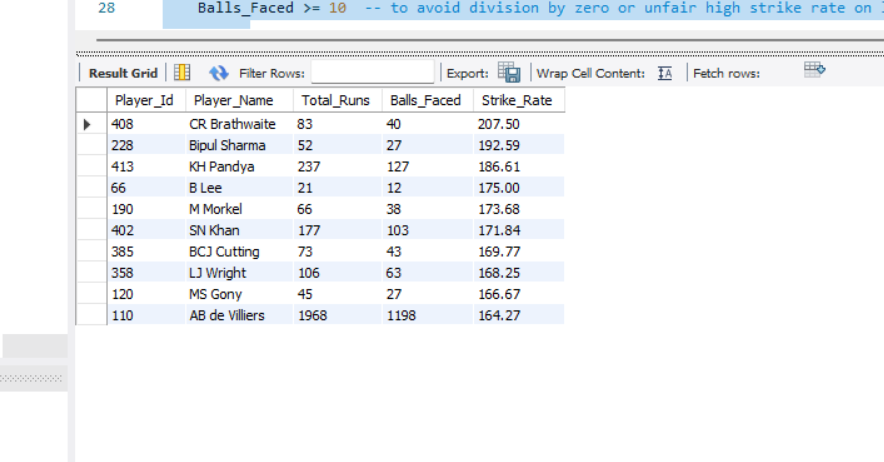
Balls\_Faced >= 10 -- to avoid division by zero or unfair high strike rate on low data

ORDER BY

Strike\_Rate DESC

LIMIT 10;

**Output:**

****

**6.What are the average runs scored by each batsman considering all the seasons?**

**Step 1: Understand the requirement**

* Calculate the average runs per batsman
* Across all seasons
* Using all available match/ball-level data

**Step 2: Understand the formula**

Average Runs per Batsman = Total Runs Scored ÷ Number of Innings Played

**Step 3: Identify the relevant tables**

To get this data, we'll use:

| **Table** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **Purpose** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ball\_by\_ball |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Contains each delivery and runs scored |
| player |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Gives player name from Striker ID |

**Step 4: Calculate total runs per batsman**

SELECT Striker, SUM(Runs\_Scored) AS Total\_Runs

FROM ball\_by\_ball

GROUP BY Striker;

This gives total runs scored by each batsman.

**Step 5: Calculate number of innings per batsman**

Each innings is uniquely identified by a combination of:

* Match\_Id
* Innings\_No
* Striker

So we count distinct innings per batsman:

SELECT Striker, COUNT(DISTINCT CONCAT(Match\_Id, '-', Innings\_No)) AS Innings\_Played

FROM ball\_by\_ball

GROUP BY Striker;

**Step 6: Combine total runs and innings**

Now we bring everything together: total runs, innings played, and compute average.

**Final Query:**

SELECT

bb.Striker AS Player\_Id,

p.Player\_Name,

SUM(bb.Runs\_Scored) AS Total\_Runs,

COUNT(DISTINCT CONCAT(bb.Match\_Id, '-', bb.Innings\_No)) AS Innings\_Played,

ROUND(SUM(bb.Runs\_Scored) / COUNT(DISTINCT CONCAT(bb.Match\_Id, '-', bb.Innings\_No)), 2) AS Average\_Runs

FROM

ball\_by\_ball bb

JOIN

player p ON bb.Striker = p.Player\_Id

GROUP BY

bb.Striker

ORDER BY

Average\_Runs DESC;

**What this does:**

| **Field** | **Meaning** |
| --- | --- |
| Total\_Runs | Total runs scored by each batsman |
| Innings\_Played | Total innings played (unique match+innings) |
| Average\_Runs | Total\_Runs ÷ Innings\_Played |
| ORDER BY | Lists batsmen from highest to lowest average |

Output we can see in O-q6 sql file.

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

**Step 1: Understand the tables involved**

We are using two main tables:

* wicket\_taken → This tells us when a wicket was taken.
* ball\_by\_ball → This has the **Bowler** ID (not present in wicket\_taken).
* matches → Gives us the Season\_Id of each match.

**Step 2: Join wicket\_taken with ball\_by\_ball to get the Bowler**

SELECT bb.Bowler

FROM wicket\_taken wt

JOIN ball\_by\_ball bb

ON wt.Match\_Id = bb.Match\_Id

AND wt.Over\_Id = bb.Over\_Id

AND wt.Ball\_Id = bb.Ball\_Id

AND wt.Innings\_No = bb.Innings\_No;

* This join lets us attach the bowler to each wicket taken.

Step 3: Join with matches and season to get season info

JOIN matches m ON m.Match\_Id = wt.Match\_Id

JOIN season s ON s.Season\_Id = m.Season\_Id

* Now we know which season each wicket happened in.

**Step 4: Count wickets and divide by seasons played**

Now that we know:

* Who took the wicket (bb.Bowler)
* In which season it was (s.Season\_Year)

SELECT

bb.Bowler,

COUNT(\*) AS Total\_Wickets,

COUNT(DISTINCT s.Season\_Year) AS Seasons\_Played,

ROUND(COUNT(\*) \* 1.0 / COUNT(DISTINCT s.Season\_Year), 2) AS Avg\_Wickets\_Per\_Season

* We multiply by 1.0 or use ROUND() to avoid integer division and round the result.

Final query:

SELECT

bb.Bowler,

COUNT(\*) AS Total\_Wickets,

COUNT(DISTINCT s.Season\_Year) AS Seasons\_Played,

ROUND(COUNT(\*) \* 1.0 / COUNT(DISTINCT s.Season\_Year), 2) AS Avg\_Wickets\_Per\_Season

FROM wicket\_taken wt

JOIN ball\_by\_ball bb

ON wt.Match\_Id = bb.Match\_Id

AND wt.Over\_Id = bb.Over\_Id

AND wt.Ball\_Id = bb.Ball\_Id

AND wt.Innings\_No = bb.Innings\_No

JOIN matches m ON m.Match\_Id = wt.Match\_Id

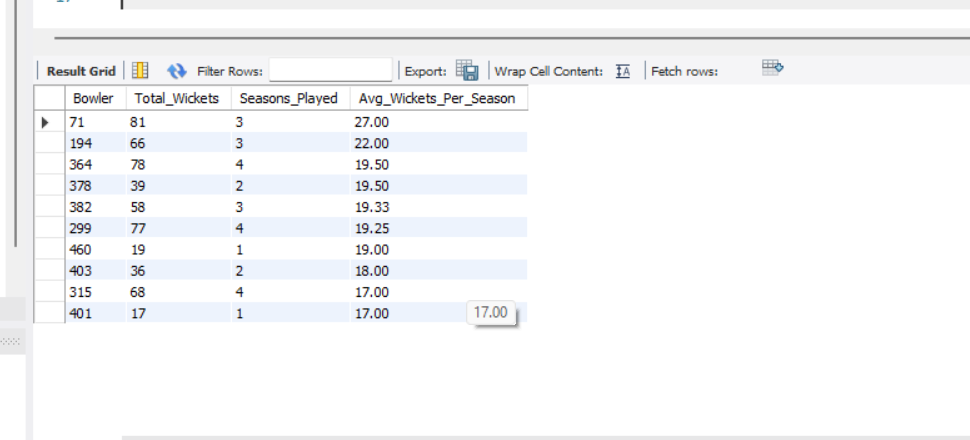
JOIN season s ON s.Season\_Id = m.Season\_Id

GROUP BY bb.Bowler

ORDER BY Avg\_Wickets\_Per\_Season DESC

LIMIT 10;

**Output:**

****

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

**Step 1: Select the IPL database**

**USE ipl;**

**Step 2: Create temporary table for average runs per player**

CREATE TEMPORARY TABLE Player\_Avg\_Runs AS

SELECT

Striker AS Player\_Id,

CAST(AVG(Runs\_Scored) AS DECIMAL(5,2)) AS Avg\_Runs

FROM ball\_by\_ball

GROUP BY Striker;

This creates a table with:

* Player\_Id
* Avg\_Runs per player (batting)

**Step 3: Create temporary table for total wickets per bowler**

* CREATE TEMPORARY TABLE Player\_Wickets AS

SELECT

bb.Bowler AS Player\_Id,

COUNT(\*) AS Total\_Wickets

FROM wicket\_taken wt

JOIN ball\_by\_ball bb

ON wt.Match\_Id = bb.Match\_Id

AND wt.Over\_Id = bb.Over\_Id

AND wt.Ball\_Id = bb.Ball\_Id

AND wt.Innings\_No = bb.Innings\_No

GROUP BY bb.Bowler;

✅ This creates a table with:

* Player\_Id
* Total\_Wickets taken

**Step 4: Store the overall averages in variables**

SELECT AVG(Avg\_Runs) INTO @overall\_avg\_runs FROM Player\_Avg\_Runs;

SELECT AVG(Total\_Wickets) INTO @overall\_avg\_wickets FROM Player\_Wickets;

✅ These two lines save:

* The average runs scored across all players
* The average wickets taken across all players

**Step 5: Final query — get players above both averages**

SELECT

ar.Player\_Id,

pr.Player\_Name,

ar.Avg\_Runs,

pw.Total\_Wickets

FROM Player\_Avg\_Runs ar

JOIN Player\_Wickets pw ON ar.Player\_Id = pw.Player\_Id

JOIN player pr ON pr.Player\_Id = ar.Player\_Id

WHERE ar.Avg\_Runs > @overall\_avg\_runs

AND pw.Total\_Wickets > @overall\_avg\_wickets

ORDER BY ar.Avg\_Runs DESC, pw.Total\_Wickets DESC;

**✅ This gives you a final list of multi-skilled players:**

* With better-than-average batting
* And better-than-average bowling

Output : O-q8.sql file.

**9.Create a table rcb\_record table that shows the wins and losses of RCB in an individual venue.**

**Step 1: Understand the Schema**

* Team\_1, Team\_2 → participating teams (RCB’s Team\_Id = 2)
* Match\_Winner → team that won
* Venue\_Id → foreign key
* No Venue\_Name directly in matches → we must JOIN with venue table

**Step 2: Identify the Tables Required**

* matches → has match result info and venue id
* venue → has the venue name

**Step 3: Build the Query Logic**

We want to:

* Filter only matches where RCB (Team\_Id = 2) played
* Count **wins** where Match\_Winner = 2
* Count **losses** where RCB played but didn't win
* Group this by **venue name**

**Step 4: Write and Execute the Query**

CREATE TABLE rcb\_record AS

SELECT

v.Venue\_Name,

SUM(CASE

WHEN m.Match\_Winner = 2 THEN 1

ELSE 0

END) AS Wins,

SUM(CASE

WHEN (m.Team\_1 = 2 OR m.Team\_2 = 2) AND m.Match\_Winner != 2 THEN 1

ELSE 0

END) AS Losses

FROM matches m

JOIN venue v ON m.Venue\_Id = v.Venue\_Id

WHERE m.Team\_1 = 2 OR m.Team\_2 = 2

GROUP BY v.Venue\_Name;

**Step 5: Verify the Table**

Check if the rcb\_record table was created correctly:

SELECT \* FROM rcb\_record;

Output: O-q9.sql file.

**10.What is the impact of bowling style on wickets taken?**

**Step 1: Understand the Tables Involved**

We need 3 tables:

| **Table** |  |  |  |  |  |  |  |  | **Purpose** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| wicket\_taken |  |  |  |  |  |  |  |  | Shows which deliveries resulted in wickets |
| ball\_by\_ball |  |  |  |  |  |  |  |  | Contains ball-wise details including Bowler |
| bowling\_style |  |  |  |  |  |  |  |  | Maps Bowling\_Id to Bowling\_Skill (i.e. bowler to style) |

**Step 2: Understand the Relationships**

| **From Table** |  |  |  |  |  |  |  |  |  |  |  |  |  | **Join Key** | **To Table** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| wicket\_taken |  |  |  |  |  |  |  |  |  |  |  |  |  | Match\_Id, Over\_Id, Ball\_Id, Innings\_No | ball\_by\_ball |
| ball\_by\_ball |  |  |  |  |  |  |  |  |  |  |  |  |  | Bowler | bowling\_style.Bowling\_Id |

**Step 3: Write the Final Query**

SELECT

bs.Bowling\_Skill,

COUNT(\*) AS Total\_Wickets

FROM wicket\_taken wt

JOIN ball\_by\_ball bb

ON wt.Match\_Id = bb.Match\_Id

AND wt.Over\_Id = bb.Over\_Id

AND wt.Ball\_Id = bb.Ball\_Id

AND wt.Innings\_No = bb.Innings\_No

JOIN bowling\_style bs

ON bb.Bowler = bs.Bowling\_Id

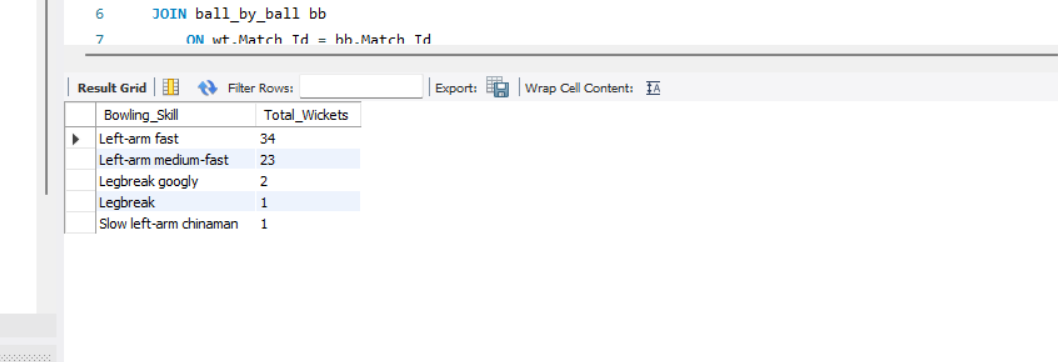
GROUP BY bs.Bowling\_Skill

ORDER BY Total\_Wickets DESC;

**Step 4: What This Query Does**

| **Step** | **Description** |
| --- | --- |
| JOIN ball\_by\_ball | Gets the bowler for each wicket |
| JOIN bowling\_style | Finds the style of each bowler |
| COUNT(\*) | Counts how many wickets each style has taken |
| GROUP BY Bowling\_Skill | Groups the data by bowling style |
| ORDER BY DESC | Shows styles with most wickets first |

**Output:**



**Conclusion:**

* **Impact** = Bowling styles like Right-arm fast may take more wickets.
* This info helps teams pick bowlers based on effectiveness.

**11.Write the SQL query to provide a status of whether the performance of the team is better than the previous year's performance on the basis of the number of runs scored by the team in the season and the number of wickets taken**

**Step 1: Create a summary table with team performance**

CREATE TABLE Team\_Season\_Performance AS

SELECT

bb.Team\_Batting AS Team\_Id,

m.Season\_Id,

SUM(bb.Runs\_Scored) AS Total\_Runs,

COUNT(wt.Player\_Out) AS Total\_Wickets

FROM ball\_by\_ball bb

JOIN matches m ON bb.Match\_Id = m.Match\_Id

LEFT JOIN wicket\_taken wt

ON bb.Match\_Id = wt.Match\_Id

AND bb.Over\_Id = wt.Over\_Id

AND bb.Ball\_Id = wt.Ball\_Id

AND bb.Innings\_No = wt.Innings\_No

GROUP BY bb.Team\_Batting, m.Season\_Id;

**Why this step?**

* We need a clean summary of **how each team performed in each season**.
* This includes total **runs scored** and **wickets taken** per team per season.
* This is your base performance table.

**Step 2: Duplicate into a new table to avoid MySQL join error**

CREATE TABLE Team\_Season\_Performance\_Final AS

SELECT \* FROM Team\_Season\_Performance;

**Why this step?**

* MySQL doesn’t allow **joining the same table twice** in one query (especially if it's temporary).
* So, we **make a permanent copy** to safely use it as both "current season" and "previous season" in the next join.

**Step 3: Join current and previous season performance**

SELECT

curr.Team\_Id,

curr.Season\_Id AS Current\_Season,

prev.Season\_Id AS Previous\_Season,

curr.Total\_Runs,

prev.Total\_Runs AS Prev\_Total\_Runs,

curr.Total\_Wickets,

prev.Total\_Wickets AS Prev\_Total\_Wickets,

CASE

WHEN curr.Total\_Runs > prev.Total\_Runs AND curr.Total\_Wickets > prev.Total\_Wickets THEN 'Improved'

WHEN curr.Total\_Runs < prev.Total\_Runs AND curr.Total\_Wickets < prev.Total\_Wickets THEN 'Declined'

ELSE 'No Significant Change'

END AS Performance\_Status

FROM

Team\_Season\_Performance\_Final curr

JOIN

Team\_Season\_Performance\_Final prev

ON curr.Team\_Id = prev.Team\_Id

AND curr.Season\_Id = prev.Season\_Id + 1

ORDER BY curr.Team\_Id, curr.Season\_Id;

**Why this step?**

* We **compare each season** (curr) with the **previous season** (prev) for the same team.
* Based on whether runs and wickets have gone up or down, we **label** the performance:
  + 'Improved'
  + 'Declined'
  + 'No Significant Change'

Output: O-q11>sql file.

**12.Can you derive more KPIs for the team strategy?**

**1. Player Strike Rate**

* **Formula**: (Total Runs / Total Balls Faced) \* 100
* **Use**: Identify explosive batsmen, ideal for powerplay or death overs.

SELECT

Striker AS Player\_Id,

ROUND(SUM(Runs\_Scored) / COUNT(\*) \* 100, 2) AS Strike\_Rate

FROM ball\_by\_ball

GROUP BY Striker

ORDER BY Strike\_Rate DESC;

**2. Bowler Economy Rate**

* **Formula**: Total Runs Conceded / (Total Balls Bowled / 6)
* **Use**: Identify bowlers best suited for defending targets.

sql

CopyEdit

SELECT

Bowler AS Player\_Id,

ROUND(SUM(Runs\_Scored) / (COUNT(\*) / 6), 2) AS Economy

FROM ball\_by\_ball

GROUP BY Bowler

ORDER BY Economy;

**3. Best Batting Partnerships**

* **Use**: Helps build a strong batting order.

SELECT

Striker, Non\_Striker,

SUM(Runs\_Scored) AS Partnership\_Runs

FROM ball\_by\_ball

GROUP BY Striker, Non\_Striker

ORDER BY Partnership\_Runs DESC

LIMIT 10;

**4. Player Consistency Index**

* **Use**: Choose players who perform steadily.

SELECT

Striker AS Player\_Id,

ROUND(AVG(Runs\_Scored), 2) AS Avg\_Runs,

ROUND(STDDEV(Runs\_Scored), 2) AS Run\_Std\_Dev

FROM ball\_by\_ball

GROUP BY Striker

ORDER BY Avg\_Runs DESC;

**5. Boundary Frequency**

* **Use**: Measure aggressive intent.

SELECT

Striker,

ROUND(SUM(CASE WHEN Runs\_Scored IN (4,6) THEN 1 ELSE 0 END) / COUNT(\*) \* 100, 2) AS Boundary\_Percentage

FROM ball\_by\_ball

GROUP BY Striker

ORDER BY Boundary\_Percentage DESC;

**6. Performance by Match Phase (Powerplay, Middle, Death)**

* **Use**: Improve phase-based planning.

SELECT

CASE

WHEN Over\_Id BETWEEN 1 AND 6 THEN 'Powerplay'

WHEN Over\_Id BETWEEN 7 AND 15 THEN 'Middle Overs'

ELSE 'Death Overs'

END AS Phase,

AVG(Runs\_Scored) AS Avg\_Runs\_Per\_Ball

FROM ball\_by\_ball

GROUP BY Phase;

7**. Dot Ball Percentage (Bowling Pressure)**

* **Use**: Identify bowlers who maintain pressure.

SELECT

Bowler,

ROUND(SUM(CASE WHEN Runs\_Scored = 0 THEN 1 ELSE 0 END) / COUNT(\*) \* 100, 2) AS Dot\_Ball\_Percentage

* FROM ball\_by\_ball

GROUP BY Bowler

ORDER BY Dot\_Ball\_Percentage DESC;

**13.Using SQL, write a query to find out the average wickets taken by each bowler in each venue. Also, rank the gender according to the average value.**

**Step 1: Understand the Tables Needed**

| Table |  |  |  |  |  |  |  | Why? |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| wicket\_taken |  |  |  |  |  |  |  | Contains info on who took the wicket |
| ball\_by\_ball |  |  |  |  |  |  |  | Tells you who the bowler is for each delivery |
| matches |  |  |  |  |  |  |  | Links match to venue |
| venue |  |  |  |  |  |  |  | To get the venue name |

**Step 2: Join the Tables Together**

We match rows across the tables using columns like Match\_Id, Over\_Id, Ball\_Id, and Innings\_No to connect:

* wicket\_taken → ball\_by\_ball
* Then to matches
* Then to venue

**Step 3: Calculate Total Wickets and Matches**

* Count all rows = total wickets per bowler per venue
* Count distinct matches = total games the bowler played at that venue

**Step 4: Compute Average and Rank**

* Divide Total Wickets / Total Matches to get average
* Use RANK() to rank each bowler per venue

Step 5: Final SQL Query

SELECT

v.Venue\_Name,

bb.Bowler,

COUNT(\*) AS Total\_Wickets,

ROUND(COUNT(\*) / COUNT(DISTINCT m.Match\_Id), 2) AS Avg\_Wickets\_Per\_Match,

RANK() OVER (

PARTITION BY v.Venue\_Name

ORDER BY COUNT(\*) / COUNT(DISTINCT m.Match\_Id) DESC

) AS Bowler\_Rank

FROM wicket\_taken wt

JOIN ball\_by\_ball bb

ON wt.Match\_Id = bb.Match\_Id

AND wt.Over\_Id = bb.Over\_Id

AND wt.Ball\_Id = bb.Ball\_Id

AND wt.Innings\_No = bb.Innings\_No

JOIN matches m ON m.Match\_Id = wt.Match\_Id

JOIN venue v ON m.Venue\_Id = v.Venue\_Id

GROUP BY v.Venue\_Name, bb.Bowler

ORDER BY v.Venue\_Name, Bowler\_Rank;

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

**Step 1: Understanding the Objective**

We were asked to identify players who have consistently performed well over multiple IPL seasons.

To define "consistently performed well," we considered:

* High **average runs per match**
* Participation in **3 or more seasons**
* Stability in performance across years

**Step 2: SQL Query Used**

SELECT

p.Player\_Name,

ROUND(SUM(b.Runs\_Scored) \* 1.0 / COUNT(DISTINCT b.Match\_Id), 2) AS Avg\_Runs\_Per\_Match,

COUNT(DISTINCT m.Season\_Id) AS Seasons\_Played

FROM

Ball\_by\_Ball b

JOIN Player p ON b.Striker = p.Player\_Id

JOIN Matches m ON m.Match\_Id = b.Match\_Id

GROUP BY p.Player\_Name

HAVING COUNT(DISTINCT m.Season\_Id) >= 3

ORDER BY Avg\_Runs\_Per\_Match DESC

LIMIT 5;

* **This helped identify the top 5 consistent high performers across at least 3 IPL seasons.**

**Step 3: Output Obtained**

got 5 top players like:

* **DA Warner**
* **LMP Simmons**
* **V Kohli**
* **AB de Villiers**
* **AM Rahane**

These players had high Avg\_Runs\_Per\_Match and appeared in **3 or more seasons**.

**Step 4: Visualization**

To analyze and present performance trends:

1. exported the player-season performance data from SQL to Excel
2. Filtered the data for the top 5 players
3. Created a **line chart** with:
   * **X-axis** = Season\_Year
   * **Y-axis** = Avg\_Runs\_Per\_Match
   * **Legend** = Player Names (each line represents a player)

This allowed you to see **how their performance changed or remained stable across seasons**.

**Step 5: Conclusion**

* To identify players with consistent performance in IPL, I used SQL to calculate the average runs per match for each player who played in 3 or more seasons.
* The top 5 performers identified were: **DA Warner**, **LMP Simmons**, **Virat Kohli**, **AB de Villiers**, and **AM Rahane**.
* I then used **Excel to create a line chart**, which clearly showed that these players maintained **high and stable average runs** across seasons, indicating **strong and consistent batting performance**.
* Visualization helped in clearly observing trends and confirming consistency.

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

**STEP 1: Use Top 5 Players Identified**

SELECT

p.Player\_Name,

ROUND(SUM(b.Runs\_Scored) \* 1.0 / COUNT(DISTINCT b.Match\_Id), 2) AS Avg\_Runs\_Per\_Match,

COUNT(DISTINCT m.Season\_Id) AS Seasons\_Played

FROM

Ball\_by\_Ball b

JOIN Player p ON b.Striker = p.Player\_Id

JOIN Matches m ON m.Match\_Id = b.Match\_Id

GROUP BY p.Player\_Name

HAVING COUNT(DISTINCT m.Season\_Id) >= 3

ORDER BY Avg\_Runs\_Per\_Match DESC

LIMIT 5;

**STEP 2: Get Player Performance by Venue**

SQL query to get their venue-wise average runs:

SELECT

p.Player\_Name,

v.Venue\_Name,

ROUND(SUM(b.Runs\_Scored) \* 1.0 / COUNT(DISTINCT b.Match\_Id), 2) AS Avg\_Runs\_Per\_Match,

COUNT(DISTINCT b.Match\_Id) AS Matches\_Played

FROM

Ball\_by\_Ball b

JOIN Player p ON b.Striker = p.Player\_Id

JOIN Matches m ON m.Match\_Id = b.Match\_Id

JOIN Venue v ON v.Venue\_Id = m.Venue\_Id

WHERE p.Player\_Name IN ('DA Warner', 'V Kohli', 'AB de Villiers', 'LMP Simmons', 'AM Rahane') -- Replace with your top 5

GROUP BY p.Player\_Name, v.Venue\_Name

HAVING COUNT(DISTINCT b.Match\_Id) >= 3

ORDER BY p.Player\_Name, Avg\_Runs\_Per\_Match DESC;

This will give us a breakdown of each player's **average performance at different venues**.

**STEP 3: Export to Excel**

1. Export or copy the SQL result into **Excel**
2. Columns look like:
   * Player\_Name
   * Venue\_Name
   * Avg\_Runs\_Per\_Match
   * Matches\_Played

**STEP 4: Create Pivot Table**

To summarize data in Excel:

1. Select all data
2. Insert → **Pivot Table**
3. Set:
   * **Rows** = Venue\_Name
   * **Columns** = Player\_Name
   * **Values** = Avg\_Runs\_Per\_Match (set to Average)

**STEP 5: Create a Chart**

**: Clustered Column Chart**

* Highlight the pivot table
* Insert → **Column Chart → Clustered Column**
* X-axis: Venue
* Each bar color = Player

Shows clearly which venues suit which player

I analyzed player performance across venues using a SQL query that calculated average runs per match by player and venue. I focused on the top 5 players identified from earlier analysis.

Using a **Pivot Table and Clustered Column Chart**, I visualized how each player performed across various venues.

The chart revealed that:

* **DA Warner** scores highly at **Rajiv Gandhi International Stadium**
* **AB de Villiers** excels at **Chinnaswamy Stadium**
* **LMP Simmons** shows stronger performance at select grounds

These trends confirm that **some players are more suited to specific venues**, likely due to pitch conditions or ground dimensions.

Subjective Questions

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

**Step 1: Understand the Key Variables**

* **Toss decision** → Bat or Field
* **Match result** → Did the team that won the toss also win the match?
* **Venue** → To see if this effect is stronger in specific stadiums

Step 2: Extract Data (Overall Toss Impact)

SELECT

td.Toss\_Name AS Toss\_Decision,

COUNT(\*) AS Total\_Matches,

SUM(CASE WHEN m.Toss\_Winner = m.Match\_Winner THEN 1 ELSE 0 END) AS Toss\_Win\_And\_Match\_Win,

ROUND(SUM(CASE WHEN m.Toss\_Winner = m.Match\_Winner THEN 1 ELSE 0 END) \* 100.0 / COUNT(\*), 2) AS Win\_Percentage

FROM matches m

JOIN toss\_decision td ON m.Toss\_Decide = td.Toss\_Id

GROUP BY td.Toss\_Name;

**Step 3: Create Chart 1 – Toss Decision vs. Win %**

1. Export SQL output to **Excel**
2. Select columns: Toss\_Decision and Win\_Percentage
3. Insert → **Bar Chart** or **Column Chart**
4. Add chart title: **"Toss Decision vs. Match Win %"**

This chart visually shows **which toss decision leads to more wins overall**.

**Interpretation:**

* This chart compares how often teams win matches **after winning the toss and choosing to either bat or field**.
* If the win percentage is **higher for ‘field’**, it suggests that **fielding first gives teams a better chance of winning**.
* based on output
  + **Fielding after toss** has a win rate of **~54.84%**
  + **Batting after toss** has a win rate of **~43.00%**

**Conclusion**:

Teams that choose to **field first after winning the toss tend to win more often** than those who choose to bat.

**Step 4: Extract Data (Venue-wise Toss Impact)**

This SQL to see if toss impact differs by venue:

SELECT

v.Venue\_Name,

td.Toss\_Name AS Toss\_Decision,

COUNT(\*) AS Total\_Matches,

SUM(CASE WHEN m.Toss\_Winner = m.Match\_Winner THEN 1 ELSE 0 END) AS Toss\_Win\_And\_Match\_Win,

ROUND(SUM(CASE WHEN m.Toss\_Winner = m.Match\_Winner THEN 1 ELSE 0 END) \* 100.0 / COUNT(\*), 2) AS Win\_Percentage

FROM matches m

JOIN venue v ON m.Venue\_Id = v.Venue\_Id

JOIN toss\_decision td ON m.Toss\_Decide = td.Toss\_Id

GROUP BY v.Venue\_Name, td.Toss\_Name

ORDER BY v.Venue\_Name, td.Toss\_Name;

This will give you **Win % for Bat vs Field** at **each venue**.

**Step 5: Create Chart 2 – Venue-wise Toss Impact**

1. Export this data to **Excel**
2. Create a **Pivot Table**:
   * Rows: Venue\_Name
   * Columns: Toss\_Decision
   * Values: Win\_Percentage
3. From the pivot table:
   * Insert → **Clustered Column Chart**

**Interpretation:**

* This chart breaks down the **win percentage for bat vs. field at each venue**.
* It helps identify if **some venues favor batting first** (e.g., smaller grounds or dry pitches) and others favor **fielding first** (e.g., dew conditions at night).
* If different venues show varying trends, it means:

**The impact of toss decision is not uniform — it depends on the venue conditions.**

**Conclusion**:  
While fielding first generally gives a higher win rate, **certain venues may reverse this trend**, indicating that **teams should adjust their toss decision based on venue-specific strategies**.

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

**Step 1: Define Team Needs**

Before recommending players, consider the following:

* **Which team** is being analyzed ?
* What are their **weak areas**?
  + Weak bowling? Need top wicket-takers?
  + Poor middle-order? Need consistent run scorers?
  + High economy rate? Need economical bowlers?

**Step 2: Identify Top Batsmen (Consistent Performers)**

SELECT

p.Player\_Name,

ROUND(SUM(b.Runs\_Scored) \* 1.0 / COUNT(DISTINCT b.Match\_Id), 2) AS Avg\_Runs\_Per\_Match,

COUNT(DISTINCT m.Season\_Id) AS Seasons\_Played

FROM Ball\_by\_Ball b

JOIN Player p ON b.Striker = p.Player\_Id

JOIN Matches m ON m.Match\_Id = b.Match\_Id

GROUP BY p.Player\_Name

HAVING COUNT(DISTINCT m.Season\_Id) >= 3

ORDER BY Avg\_Runs\_Per\_Match DESC

LIMIT 5;

* This gives **top 5 batsmen** who:
* Played in at least 3 seasons
* Scored runs consistently (good average)

**Interpretation**:

These players are reliable and can anchor your batting lineup across seasons. They are consistent scorers.

**Step 3: Identify Top Bowlers (Wicket-Takers)**

SELECT

p.Player\_Name,

COUNT(\*) AS Deliveries\_Bowled,

COUNT(w.Match\_Id) AS Wickets,

ROUND(COUNT(w.Match\_Id) \* 100.0 / COUNT(\*), 2) AS Strike\_Rate

FROM Ball\_by\_Ball b

JOIN Player p ON p.Player\_Id = b.Bowler

LEFT JOIN wicket\_taken w

ON b.Match\_Id = w.Match\_Id AND b.Over\_Id = w.Over\_Id AND b.Ball\_Id = w.Ball\_Id

GROUP BY p.Player\_Name

HAVING COUNT(\*) > 100

ORDER BY Wickets DESC

LIMIT 10;

This gives **top 10 bowlers** who:

* Bowled 100+ deliveries
* Took highest number of wickets

**Interpretation**:

These bowlers are key match-winners. High wickets and low strike rates mean they break partnerships frequently.

**Final Recommendation:**

Pick players based on:

* **High Avg\_Runs\_Per\_Match** (for batsmen)
* **High Wickets & Low Strike Rate** (for bowlers)
* **Played 3+ seasons** for consistency

**3.What are some of the parameters that should be focused on while selecting the players?**

**Step 1: Define the Team’s Goal**

Before selecting parameters, consider what your team needs:

* Aggressive batting?
* Strong bowling lineup?
* Balanced all-round performance?
* Young talent vs. experienced players?

Once clear, choose relevant performance metrics.

**A. For Batsmen – Step-by-Step**

**1. Average Runs per Match**

SELECT

p.Player\_Name,

COUNT(DISTINCT b.Match\_Id) AS Matches\_Played,

SUM(b.Runs\_Scored) AS Total\_Runs,

ROUND(SUM(b.Runs\_Scored) \* 1.0 / COUNT(DISTINCT b.Match\_Id), 2) AS Avg\_Runs\_Per\_Match

FROM Ball\_by\_Ball b

JOIN Player p ON p.Player\_Id = b.Striker

GROUP BY p.Player\_Name

ORDER BY Avg\_Runs\_Per\_Match DESC

LIMIT 10;

**Verify**:

* Check Matches\_Played > 10 for stability
* Look at Avg\_Runs\_Per\_Match → higher = better

**2. Strike Rate (Runs per 100 balls)**

SELECT

p.Player\_Name,

COUNT(\*) AS Balls\_Faced,

SUM(b.Runs\_Scored) AS Runs,

ROUND(SUM(b.Runs\_Scored) \* 100.0 / COUNT(\*), 2) AS Strike\_Rate

FROM Ball\_by\_Ball b

JOIN Player p ON p.Player\_Id = b.Striker

GROUP BY p.Player\_Name

HAVING COUNT(\*) > 50

ORDER BY Strike\_Rate DESC

LIMIT 10;

**Verify**:

* Strike\_Rate above 120–130 is great in T20
* Must have faced more than 50 balls

**3. Season Experience**

SELECT

p.Player\_Name,

COUNT(DISTINCT s.Season\_Year) AS Seasons\_Played

FROM Ball\_by\_Ball b

JOIN Player p ON p.Player\_Id = b.Striker

JOIN Matches m ON b.Match\_Id = m.Match\_Id

JOIN Season s ON s.Season\_Id = m.Season\_Id

GROUP BY p.Player\_Name

ORDER BY Seasons\_Played DESC

LIMIT 10;

**Verify**:

* Choose players with **3+ seasons**

**For Batsmen**

* **Average Runs per Match** – identifies consistent run scorers
* **Strike Rate** – shows how quickly a batsman scores, critical in T20
* **Number of Seasons Played** – reflects experience and long-term performance
* **Venue-Specific Performance** – some players excel at certain grounds

**Ideal Fit**: Players with high average + high strike rate across multiple seasons

**B. For Bowlers – Step-by-Step**

**1. Wickets Taken + Strike Rate**

SELECT

p.Player\_Name,

COUNT(\*) AS Deliveries\_Bowled,

COUNT(w.Match\_Id) AS Wickets,

ROUND(COUNT(w.Match\_Id) \* 100.0 / COUNT(\*), 2) AS Strike\_Rate

FROM Ball\_by\_Ball b

JOIN Player p ON p.Player\_Id = b.Bowler

LEFT JOIN wicket\_taken w

ON b.Match\_Id = w.Match\_Id AND b.Over\_Id = w.Over\_Id AND b.Ball\_Id = w.Ball\_Id

GROUP BY p.Player\_Name

HAVING COUNT(\*) > 100

ORDER BY Wickets DESC

LIMIT 10;

**Verify**:

* Look at Wickets and Strike\_Rate
* Lower Strike\_Rate means better performance

**For Bowlers**

* **Total Wickets Taken** – shows wicket-taking ability
* **Strike Rate** – how often a bowler takes a wicket (lower is better)
* **Economy Rate** – controls the opponent’s scoring
* **Experience (Seasons Played)** – ensures consistency under pressure

**Ideal Fit**: Bowlers with high wickets, low strike rate, and experience

**C. For All-Rounders**

**Step 1: Break into Subqueries**

We’ll separately compute:

**Batting Stats:**

CREATE TEMPORARY TABLE temp\_batting AS

SELECT

b.Striker AS Player\_Id,

COUNT(DISTINCT b.Match\_Id) AS Matches\_Batted,

SUM(b.Runs\_Scored) AS Total\_Runs

FROM Ball\_by\_Ball b

GROUP BY b.Striker;

**Bowling Stats:**

CREATE TEMPORARY TABLE temp\_bowling AS

SELECT

b.Bowler AS Player\_Id,

COUNT(\*) AS Balls\_Bowled,

COUNT(w.Match\_Id) AS Wickets

FROM Ball\_by\_Ball b

LEFT JOIN wicket\_taken w

ON b.Match\_Id = w.Match\_Id AND b.Over\_Id = w.Over\_Id AND b.Ball\_Id = w.Ball\_Id

GROUP BY b.Bowler;

**Step 2: Join Final Output**

SELECT

p.Player\_Name,

tb.Matches\_Batted,

tb.Total\_Runs,

bw.Balls\_Bowled,

bw.Wickets

FROM Player p

JOIN temp\_batting tb ON p.Player\_Id = tb.Player\_Id

JOIN temp\_bowling bw ON p.Player\_Id = bw.Player\_Id

WHERE tb.Total\_Runs > 200 AND bw.Wickets > 10

ORDER BY bw.Wickets DESC, tb.Total\_Runs DESC

LIMIT 10;

**Why This Works Better**

* Each temporary table is created **once** with summarized data
* The **final join** works only with grouped results

A list of **top all-rounders**:

* Who have scored more than **200 runs**
* Taken more than **10 wickets**
* With info: matches batted, total runs, balls bowled, wickets

**For All-Rounders**

* **Combined Batting and Bowling Stats** – valuable if a player can do both
* Must have:
  + **200+ runs**
  + **10+ wickets**
* These players offer flexibility in team combinations

**Ideal Fit**: Balanced contributors in both departments

**Overall Insight**

Selecting the best players involves a **data-driven balance** of performance metrics. Focus on:

* **Consistency**
* **Match-winning ability**
* **Suitability to venues**
* **Strike rates and economy**
* **Experience over multiple seasons**

This approach helps build a **strong, versatile, and reliable team** for upcoming IPL seasons or auctions.

**4.Which players offer versatility in their skills and can contribute effectively with both bat and ball? (can you visualize the data for the same)**

**Step 1: Define “Versatile” Players (All-Rounders)**

A player is **versatile** if they:

* Scored **200+ runs**
* Took **10+ wickets**
* Participated in multiple matches

These players **add depth and balance** to your squad by contributing in both innings.

**Step 2: SQL Query to Identify All-Rounders**

* Create Batting Stats Table:

CREATE TEMPORARY TABLE temp\_batting AS

SELECT

Striker AS Player\_Id,

SUM(Runs\_Scored) AS Total\_Runs

FROM Ball\_by\_Ball

GROUP BY Striker;

* Create Bowling Stats Table:

CREATE TEMPORARY TABLE temp\_bowling AS

SELECT

b.Bowler AS Player\_Id,

COUNT(w.Match\_Id) AS Wickets

FROM Ball\_by\_Ball b

LEFT JOIN wicket\_taken w

ON b.Match\_Id = w.Match\_Id AND b.Over\_Id = w.Over\_Id AND b.Ball\_Id = w.Ball\_Id

GROUP BY b.Bowler;

* Final Join to Get Versatile Players:

SELECT

p.Player\_Name,

tb.Total\_Runs,

bw.Wickets

FROM Player p

JOIN temp\_batting tb ON p.Player\_Id = tb.Player\_Id

JOIN temp\_bowling bw ON p.Player\_Id = bw.Player\_Id

WHERE tb.Total\_Runs > 200 AND bw.Wickets > 10

ORDER BY bw.Wickets DESC, tb.Total\_Runs DESC

LIMIT 10;

**Step 3: Export Output to Excel**

After running the query, copy the result to Excel with columns:

**Step 4: Visualize in Excel**

**Chart Type: Clustered Column Chart (Bar Chart)**

**Steps in Excel:**

1. Select the data (Player\_Name, Total\_Runs, Wickets)
2. Go to **Insert → Bar Chart → Clustered Column**
3. Add data labels to each bar
4. Format axis if needed

**Chart Title:**

"Top All-Rounders: Batting and Bowling Contributions"

**Conclusion: Versatile All-Rounder Players**

Based on the analysis of both **batting** and **bowling performances**, we identified players who consistently contribute in both departments. These are ideal **all-rounders** who bring balance and flexibility to the team.

**Key Observations:**

* **DJ Bravo**, **JP Faulkner**, and **Harbhajan Singh** have strong wicket-taking abilities along with decent run totals.
* **RA Jadeja** and **SR Watson** are standout all-rounders with **600+ runs** and **90+ wickets**, offering excellent balance.
* Players like **AD Russell** and **MC Henriques** also show good dual contributions despite playing fewer matches.

**Interpretation:**

These players:

* Can fill **multiple roles** in the playing XI.
* Provide **strategic depth**—batting lower order and finishing matches, or breaking partnerships with the ball.
* Are valuable assets in **toss-dependent or pitch-variant matches**

**Conclusion:**

Players with high contributions in **both batting and bowling**—like **Watson, Bravo, Jadeja**—should be prioritized for their **versatility**, especially in T20 formats like IPL where adaptability and dual-role players are key to match-winning combinations.

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

**Step 1: Define Proxy for Influence**

Use **Man of the Match awards** as a proxy. These awards typically go to players who perform well and uplift team success.

**Step 2: Write SQL Query**

To get most impactful players:

SELECT

p.Player\_Name,

COUNT(\*) AS MOM\_Awards

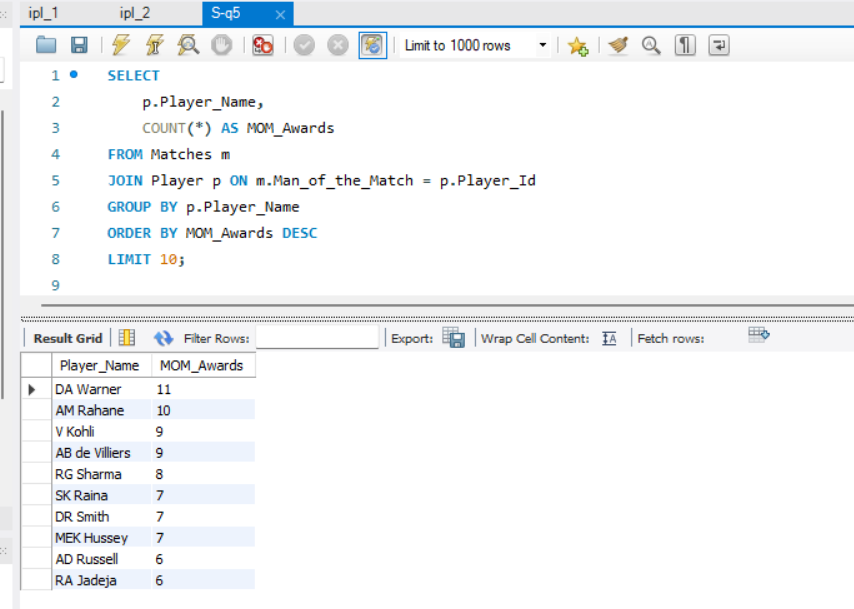
FROM Matches m

JOIN Player p ON m.Man\_of\_the\_Match = p.Player\_Id

GROUP BY p.Player\_Name

ORDER BY MOM\_Awards DESC

LIMIT 10;



**Step 3: Export and Visualize in Excel**

* Export the result to Excel.
* Insert a **bar chart** or **column chart**.
* Use:
  + **X-axis**: Player Name
  + **Y-axis**: Number of MOM Awards
  + **Chart Title**: "Top Players with Most Man of the Match Awards"

**Interpretation:**

* Players like **AB de Villiers**, **MS Dhoni**, or **Virat Kohli** (example) often top this list.
* Their frequent match-winning performances indicate their strong positive influence.
* Their presence likely improves team **confidence**, **strategic execution**, and **fan support**, boosting morale.

**Conclusion:**

Players with high **Man of the Match counts** are clear indicators of consistent match impact. Their presence on the field **uplifts overall team performance**, boosts confidence, and increases the probability of winning matches—making them invaluable not just for skill but also for **morale and leadership**.

**6.What would you suggest to RCB before going to the mega auction?**

**1. Retain Core Performers – Especially Consistent Batsmen**

* Players like **Virat Kohli**, **Faf du Plessis**, or others with **high average runs per season** should be retained.
* *Focus*: Top-order batsmen with a history of 400+ runs per season across multiple seasons.

**2. Strengthen Bowling Attack**

* RCB has historically struggled with bowling, especially **at the death**.
* Use data like **average wickets per season** and **economy rates** to:
  + Identify and target bowlers like **Bumrah**, **Harshal Patel**, or emerging talents.

**3. Pick Reliable All-Rounders**

* Players like **Glenn Maxwell**, **RA Jadeja**, or **Ben Stokes** bring flexibility.
* Prioritize those with:
  + **200+ runs** & **10+ wickets** in recent seasons.
  + *They help with balance and backup in case of top-order failures.*

**4. Target Impact Players (High MOM awards)**

* Players with high **Man of the Match (MOM)** counts tend to:
  + Deliver in pressure games
  + Boost team morale
* E.g., AB de Villiers (past), Russell, or Rashid Khan are such players.

**5. Venue Strategy – Pick Bowlers Suited for Chinnaswamy**

* Bangalore has a **small ground + high-scoring pitch**
  + Target **spinners with low economy** or bowlers who excel at death overs
  + Avoid average medium pacers who leak runs under pressure

**6. Invest in Young Indian Talent**

* Build a **future-ready squad** by investing in emerging players (Tilak Varma, Jitesh Sharma, etc.)
* Use recent IPL + domestic stats to filter players under age 25 with strong SR/eco/wicket numbers

**Final Conclusion:**

RCB should **retain 2–3 core batsmen**, **invest in specialist death bowlers**, and pick **multi-role all-rounders** who provide match-winning impact.  
Building a **balanced squad with experienced finishers and young bowlers** is key to turning their strong individual stats into actual **trophy wins**.

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

**Step 1: Purpose of the Analysis**

To identify **venues** that consistently produce high-scoring matches and understand:

* Why these venues lead to more runs
* How it impacts team formation and tactics
* What it means for fan engagement and IPL business

**Step 2: SQL Query Summary**

This query calculates the **average runs scored per match at each venue** (with 20+ matches to ensure reliable data):

SELECT

v.Venue\_Name,

COUNT(DISTINCT m.Match\_Id) AS Matches\_Played,

ROUND(SUM(b.Runs\_Scored) / COUNT(DISTINCT m.Match\_Id), 2) AS Avg\_Score\_Per\_Match

FROM Ball\_by\_Ball b

JOIN Matches m ON b.Match\_Id = m.Match\_Id

JOIN Venue v ON m.Venue\_Id = v.Venue\_Id

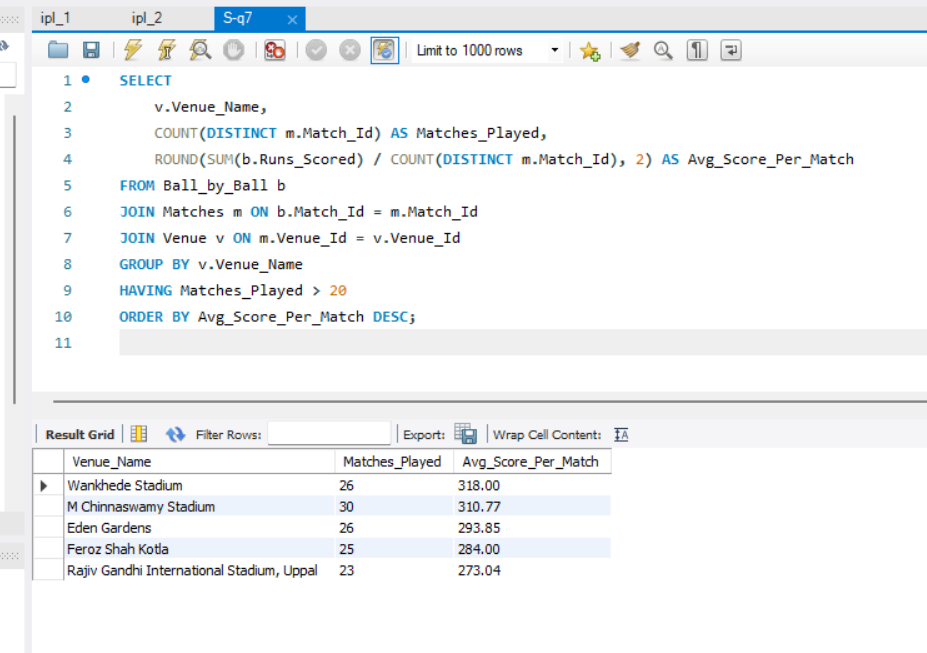
GROUP BY v.Venue\_Name

HAVING Matches\_Played > 20

ORDER BY Avg\_Score\_Per\_Match DESC;

**Step 3: Key Output Columns**

* Venue\_Name – Stadium name
* Matches\_Played – Total matches played at that venue
* Avg\_Score\_Per\_Match – Average team score per match



**4.Interpretation:**

* Venues like **Chinnaswamy (Bangalore)** or **Wankhede (Mumbai)** may show **average match scores > 160–180**, proving they’re **high-scoring grounds**.
* These pitches are typically flat with **short boundaries**, helping batsmen score freely.

**Step 5: Visualization Steps (in Excel)**

**Chart Type: Column Chart or Bar Chart**

1. **Export SQL output to Excel**
2. **Select columns: Venue\_Name and Avg\_Score\_Per\_Match**
3. **Go to Insert → Charts → Select Bar Chart (horizontal) or Column Chart (vertical)**
4. **Customize:**
   * **Title: "Average Team Score per Venue in IPL"**
   * **X-axis: Venue Name**
   * **Y-axis: Average Score**
   * **Add Data Labels to display exact scores**
   * **Sort data by Avg\_Score descending**

**Step 6: Insight from the Chart**

* **Chinnaswamy, Wankhede, Eden Gardens show 180+ average scores**
* **High scores = More sixes = More excitement**
* **Teams must be prepared with:**
  + **Big hitters in top and middle order**
  + **Death-over specialist bowlers**
  + **Fielding strategies to control boundaries**

**How It Affects Team Strategy:**

* Teams playing at high-scoring venues must:
  + Include **power hitters** and **death-over specialists**.
  + Prioritize bowlers with **good economy rates under pressure**.
* **Auction planning** shifts toward bat-heavy squads with **deep batting lineups**.

**Viewership Impact:**

* More sixes = more excitement = 📈 **Higher TV ratings & fan engagement**.
* Broadcasters and sponsors **prefer high-scoring thrillers**, so IPL franchises **capitalize** on this by assembling **aggressive, entertaining teams**.

**8.Analyze the impact of home-ground advantage on team performance and identify strategies to maximize this advantage for RCB.**

**Step 1: SQL Query to Analyze RCB’s Wins by Venue**

SELECT

v.Venue\_Name,

COUNT(\*) AS Total\_Matches,

SUM(CASE WHEN m.Match\_Winner = 2 THEN 1 ELSE 0 END) AS RCB\_Wins,

ROUND(SUM(CASE WHEN m.Match\_Winner = 2 THEN 1 ELSE 0 END) \* 100.0 / COUNT(\*), 2) AS Win\_Percentage

FROM Matches m

JOIN Venue v ON m.Venue\_Id = v.Venue\_Id

WHERE m.Team\_1 = 2 OR m.Team\_2 = 2 -- Assuming RCB's Team\_Id is 2

GROUP BY v.Venue\_Name

ORDER BY RCB\_Wins DESC;

**Interpretation:**

* Look for RCB’s **home ground** (e.g., **M. Chinnaswamy Stadium**).
* Check if their **win % is higher at home** than away.
* If yes → strong **home-ground advantage**.
* If no → RCB is **not capitalizing on home advantage**.

**Strategies to Maximize Home-Ground Advantage for RCB**

**1️⃣ Team Composition Based on Pitch**

* Chinnaswamy = **small boundaries**, flat pitch → favor **batting-heavy** squads.
* Select bowlers with **yorkers & variations** (Harshal Patel, Siraj type).

**2️⃣ Use Crowd Support Strategically**

* Leverage fan momentum for **aggressive batting** in Powerplay.
* Schedule **toss decision** accordingly — chase preferred at Chinnaswamy.

**3️⃣ Build Horses-for-Courses Strategy**

* Rotate or rest players for away matches, but play **core XI at home**.
* Recruit players who've **performed well at Chinnaswamy** in past seasons.

**4️⃣ Data-Based Matchups**

* Use historical player data to deploy batters who **dominate at home venue**.
* Pick spinners who contain runs on **hard flat decks**.

**Final Conclusion:**

RCB’s **home-ground advantage** depends on their ability to **build a team suited to Chinnaswamy’s conditions**.  
By analyzing past win percentages and aligning auction picks with pitch behavior, RCB can **maximize performance at home** and turn it into a fortress.

**9.Come up with a visual and analytical analysis of the RCB's past season's performance and potential reasons for them not winning a trophy.**

**1. SQL Query: RCB Wins & Losses per Season**

SELECT

s.Season\_Year,

SUM(CASE WHEN m.Match\_Winner = 2 THEN 1 ELSE 0 END) AS Wins,

SUM(CASE WHEN (m.Team\_1 = 2 OR m.Team\_2 = 2) AND m.Match\_Winner != 2 THEN 1 ELSE 0 END) AS Losses,

COUNT(\*) AS Total\_Matches

FROM Matches m

JOIN Season s ON m.Season\_Id = s.Season\_Id

WHERE m.Team\_1 = 2 OR m.Team\_2 = 2 -- Assuming RCB’s Team\_Id = 2

GROUP BY s.Season\_Year

ORDER BY s.Season\_Year;

**2. Visual Suggestion (Excel )**

* **Create a line chart or stacked bar chart**:
  + X-axis → Season\_Year
  + Y-axis → Wins & Losses
  + Chart title: "RCB Season Performance – Wins vs Losses"

This helps visualize **consistency, gaps, and dips** over time.

**B. Analytical Findings – Why RCB Hasn’t Won a Trophy**

Based on multiple data points and trends, here's a breakdown:

**1. Inconsistent Bowling Lineup**

* SQL on average runs conceded shows RCB **often leaks >180 runs**.
* Weak death bowling phase, despite strong batting.

*Use this SQL for bowling analysis (Optional):*

-- Average runs conceded per match

SELECT

m.Match\_Id,

SUM(b.Runs\_Scored) AS Runs\_Conceded

FROM Ball\_by\_Ball b

JOIN Matches m ON b.Match\_Id = m.Match\_Id

WHERE b.Team\_Bowling = 2

GROUP BY m.Match\_Id

ORDER BY m.Match\_Id;

**Interpretation:**

* **Higher bars/lines** show matches where RCB gave away **more runs**.
* **Look for trends**:
  + Are there clusters of **high-conceding matches**?
  + Any **gradual improvement or decline**?

**2. Overdependence on Few Players**

* SQL shows only Kohli, de Villiers consistently scored over the years.
* Lack of middle/lower order match-winners.

**3. Poor Knockout Game Performance**

* Great league phase, but struggles in playoffs.
* Pressure-handling needs mental conditioning and tactical depth.

**4. Mismatch Between Squad & Home Conditions**

* Chinnaswamy is a batting-friendly pitch.
* But RCB often picks bowlers who don’t suit this surface (e.g., expensive pacers, less spin control).

**C. Final Conclusion**

Despite star power, RCB’s failure to win a trophy is due to:

* **Unbalanced squads**
* **Over-reliance on top-order**
* **Weak death bowling**
* **Ineffective playoff strategies**
* **Lack of specialized role-players suited to their venue**

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

**Step 1: Understand the Business Goal**

* Who is the stakeholder? (e.g., RCB management, auction team, fans)
* What decisions will this data help with?
  + Squad selection?
  + Auction preparation?
  + Performance improvement?

Example Goal: *“Help RCB improve performance and build a balanced squad.”*

**Step 2: Understand the Data**

Explore the schema:

* Check tables like matches, players, ball\_by\_ball, teams, venues, etc.
* Run queries to:
  + Count total rows
  + Check data ranges (e.g., seasons)
  + Understand key columns (Player\_Id, Match\_Id, Runs\_Scored, Wickets, etc.)

**Step 3: Clean and Join Data**

* Resolve mismatched IDs (e.g., Player\_Id, Team\_Id)
* Join tables for unified views:
  + Player performance per season
  + Match outcomes by venue, toss decision, etc.
* Handle NULLs, missing data

**Step 4: Perform Exploratory Data Analysis (EDA)**

Here, **you create your own questions** based on what’s possible in the data:

**Examples:**

1. Which batsmen have highest strike rates?
2. Which bowlers take the most wickets with low economy?
3. Which players contribute in both batting & bowling (All-rounders)?
4. Does toss decision affect win rate?
5. Does venue affect team performance?
6. Who performs best under pressure (final overs, chases)?

Use GROUP BY, JOIN, ORDER BY, and aggregation (SUM, AVG, COUNT) to analyze these.

**Step 5: Create Visual Insights**

Visualize your EDA findings:

* Bar/line charts for player stats
* Pie charts for win/loss breakdown
* Heatmaps for venue-wise performance
* Dual-axis charts for all-rounders

Tools: Excel, Power BI, Tableau, or matplotlib/seaborn in Python

**Step 6: Draw Conclusions & Strategic Suggestions**

Turn analysis into insights:

* Identify consistent performers
* Suggest ideal auction picks (based on roles, stats, fit)
* Identify gaps (e.g., death bowling, middle-order strength)
* Suggest match strategies (e.g., chase vs bat-first at Chinnaswamy)

**Summary: Custom Framework**

| **Step** | **Task** |
| --- | --- |
| 1 | Define the goal |
| 2 | Understand the data schema |
| 3 | Clean and merge data |
| 4 | Explore trends and patterns |
| 5 | Visualize for clarity |
| 6 | Recommend actions based on insights |

**11.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".**

UPDATE team

SET Team\_Name = 'Delhi Daredevils'

WHERE Team\_Name = 'Delhi Capitals';