IPL 2008 Dataset

Matches Dataset (matches.csv)

Row count : 577 rows

Column count : 18 columns

- **13 unique teams.**
- **9 IPL seasons.**
- **30 different cities.**

Missing values:-

City – 7 values
Winner – 3 values
Player of match – 3 values – matches without clear reasult
Umpire 3 - empty column

Columns:-

1. **id:** (Int) - Primary key - A unique numeric identifier for each match.
2. **season: (Year (DateTime)** The year or season when the match was played (e.g., 2008, 2009, ..., 2016).
3. **city:** (Varchar) The city in which the match was played.
4. **Date**: (Date Time) The calendar date on which the match took place.
5. **Team1:** (Varchar) The first team listed as playing in the match.
6. **team2:** (Varchar) The second team listed in the match.
7. **toss_winner:** (Varchar) The team that won the toss at the beginning of the match.
8. **toss_decision:** (Enum) The decision (usually "bat" or "field") made by the toss-winning team.
9. **result:** (Enum) The type of result for the match ["normal", "tie", "no result"]
   Important when filtering out matches for certain analyses (e.g., only considering "normal" results).
10. **dl_applied: (Boolean)** Indicator (0 or 1) showing whether the Duckworth-Lewis method was applied.
    Signifies if a match was affected by rain or other interruptions.
    Helps in understanding matches that might have an atypical progression or result.
11. **winner:** (Varchar) The team that won the match.
    Central to outcome-based analysis.
    Used for calculating win percentages and performance comparisons among teams.
12. **win_by_runs: (Int)** The margin of victory in terms of runs (applicable when the team batting first wins).

13. **win_by_wickets: (Int)** The margin of victory in terms of wickets (applicable when the chasing team wins).
14. **player_of_match**: (Varchar)The player standout performer in the match.
15. **venue:** (Varchar) The stadium or ground where the match was held. Different venues may have different pitch characteristics, influencing game outcomes.
16. **umpire1**: (Varchar) Name of the first umpire officiating the match.
17. **umpire2**: (Varchar) Name of the second umpire.
18. **umpire3**: (Varchar) Name of the third umpire (usually involved in video review decisions).

II. Deliveries Dataset (deliveries.csv)

Row Count : 136,598

Column Count : 21

13 teams, 334 bowlers, 436 batsmen.

Missing Values :-

In the deliveries table, missing values appear in columns related to dismissals. Specifically, the **"player_dismissed"** and **"dismissal_kind"** columns are missing in **129,871** rows out of **136,598** because no wicket fell on most deliveries. Similarly, the "**fielder**" column is missing in **131,727** rows, again **because it only gets populated when there is a dismissal that involves a fielder.**

Columns:-

1. **match_id:** (Primary key of matches) A numeric identifier that links the delivery to a specific match in matches.csv. (Foreign key)
   Essential for joining ball-by-ball data with match-level data.
   Ensures that analysis of individual deliveries can be related to match outcomes.

2. **inning:** (Enum) Indicates the inning number [1,2] in which the delivery was bowled.

3. **batting_team**: (Varchar) The name of the team that is batting during that delivery.

4. **bowling_team: (Varchar)** The name of the team that is bowling during that delivery.

5. **Over : (Int 1 to 20)** The over number during which the delivery is bowled.
6. **Ball: (Int 1 to 6)** The ball number within the over

7. **Batsman: (Varchar)** The player facing the delivery.
8. **non_striker: (Varchar)** The batsman at the other end who is not facing the delivery.
9. **Bowler: (Varchar)** The bowler delivering the ball.
10. **is_super_over: (Boolean)** A flag (0 or 1) indicating whether the delivery is part of a super over (used in tie-break situations).
11. **wide_runs: (Int)** Runs awarded due to a wide ball (illegal delivery wide of the batsman).
12. **bye_runs: (Int)** Runs taken as byes (when the ball passes the batsman without touching the bat and runs are taken).
13. **legbye_runs: (Int)** Runs taken as leg byes (runs taken when the ball hits the batsman's body rather than the bat).
14. **noball_runs: (Int)** Runs awarded for a no-ball (illegal delivery usually due to overstepping).
    Results in an extra delivery and extra runs.
15. **penalty_runs: (Int)** Additional runs awarded as a penalty for certain rule infringements.
16. **batsman_runs: (Int)** Runs scored directly off the bat by the batsman on that delivery.
17. **extra_runs : (Int)** Total extra runs conceded on the delivery (sum of wides, byes, leg byes, no-balls, and penalties).
18. **total_runs: (Int)** The sum of batsman_runs and extra_runs on the delivery.
19. **player_dismissed: (Varchar)** The name of the player dismissed as a result of that delivery (if any).
20. **dismissal_kind: (Enum)** The method by which the dismissal occurred (bowled, caught, lbw, run out).
21. **Fielder: (Varchar)** The name of the fielder involved in the dismissal (who caught the ball).

1. How to create a unique key if ID not present? How will you join the two data sets?

-> Asume both tables dont have any id column

-- Sort the matches data on match_date
-- create match_id column auto increament column

Now to perform join we need match_id in deliveries also
-- Now create some new columns as
current_inning , previous_inning , match_id
-- Count=1

```
For i in range(len(data)) :
        Current_inning = inning
        Previous_Inning = previous value of  (current inning)  // can use sql function to find
                                                                  previous value of column

        If ( current_inning == 1 AND previous_Inning == 2 ):
        Count+=1;
        Match_id=count
        continue
If else :
match_id=count;
continue
```

In this way match id will be there in both tables hence join can be performed on match_id

2. Which Bowler in the IPL has taken maximum wickets?

-> Will take rows where dissmissle not In (‘run _ out ‘ , ‘retired’ , ‘hurt’ , ‘obstructing_field’)
Group by bowler and count (bowler)
Max_wicket = max(count(bowler))

3 .What is the strike rate of BB McCullum in IPL 2008?

-> Join both tables on match id
-- mergedf = select rows where batsman == BB McCullum and season ==2008
-- Filter_df= mergedf [(mergdf['no_ball_runs']==0) & (mergedf['wide_runs']== 0)]

-- Total_runs=filte_df['batsman_run'] .sum()
-- Total_balls= len(filter_df) -- only row where season is 2008 and player is BB McCullum
And runs collected not include wide runs or no ball run

-- Strike_rate=(total_runs/total_balls)

4. Which IPL Team concedes maximum extra runs?

-> group by bowling team with sum of extra run

-- Max_extra_run_df = deliveries_df. Groupby('bowling_team')['extra_run'].sum()

-- Now sort te max extra run df in descending order

-- 1st one will be the team conceding maximum extra runs