Indian premiere league

Sanskar Pareta(2017CS10373),Kailash Kumawat(2017CS10338)

March 2, 2020

1 SECTION 1

1.1 INTRODUCTION

Our project is covering many things that an IPL fan wants to know between 2008 and 2017. There are pre-existing sites that give a lot of detail about everything, but our work is covering small gaps that don't require the user to type anything here, just a short name and results within a second have been shown. We have a limited data set but we have tried to serve the user at our best level. We have created a project where you can see all the statistics of IPL till 2017. User can search any player and get statistics and profile of that player. Users can also get information about the team like the total matches played by any team in any season and also the matches won by them in each season, you can get the list of players who played by each player. Have played for that team in order of matches. The user can get information about different grounds, where an ipl match has been played with some information or statistics about that ground. The user is also allowed to apply some filters, such as country and season, to get information about the players coming after that filter.

1.2 ER Diagram

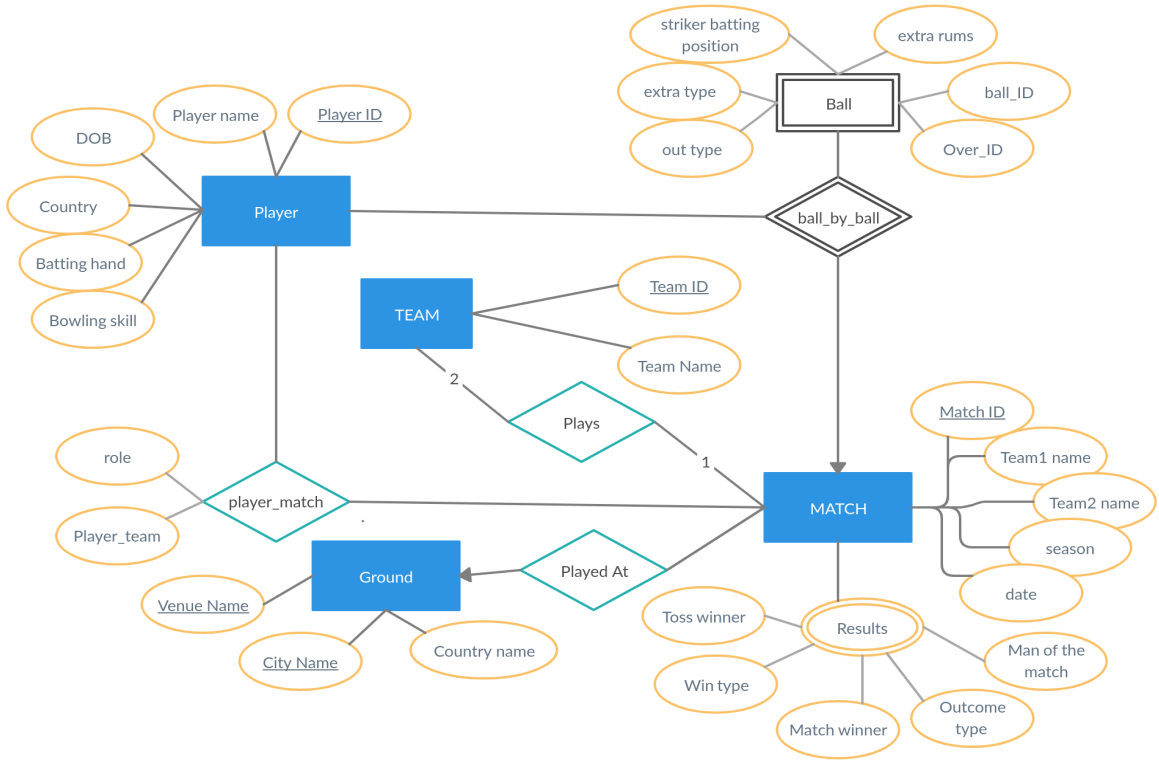

Figure 1: ER Diagram

1.3 Entities and their Attributes

Entity	*Attribute*
Team	team_id,team_name
Player	(player_id, player_name, dob, batting_hand, bowling_skill, country_name)
Ground	(venue_name, city_name, country_name)
Match	(match_id, team1, team2, match_date, season_year, venue_id, toss_winner,match_winner, toss_name, win_type, outcome_type, man_of_match, win_margin)
Player_match	(match_id, player_id, role_desc, player_team)
Match_by_team	(match_id, innings_no, team_batting, team_bowling)
Ball_by_ball	(match_id, over_id, ball_id, innings_no, striker_batting_position, extra_type,runs_scored, extra_runs, out_type, striker, non_striker, bowler, player_out,fielder)

2 Section 2

2.1 Data Source

We downloaded our dataset from 'https://www.kaggle.com/raghu07/ipl-data-till-2017' We downloaded it Readymade from the source but made some changes to it as described below. Our raw dataset has 5 tables named Team, Player, Match, Player_match and ball_by_ball.

2.2 Data Cleaning

Steps to clean data

1. There are some spelling mistakes in the data which we corrected by libreOffice.

2. There were some entries in column which do not match with other entries of that column like in one of the table, in the winning team of column like in the team column at some places there are team id's but at some place name of teams were given so we used libre office to remove this type of inconsistencies as well.

3. Dataset has some column which are surrogate keys which are produced by the person who uploaded it so we remove all that columns that are ending with sk.

4. Dataset has some anomalies like it has some column which has data that is either unnecessary for us or either can be computed from other columns so we removed them by using queries like we changed ball by ball in the way written below:

CREATE TABLE temp2 table AS SELECT match_id, over_id, ball_id, innings_no, striker_batting_position, extra_type,runs_scored, extra_runs, out_type, striker, non_striker, bowler, player_out,fielders as fielder,Bowler_Wicket from ball_by_ball;

DROP table ball_by_ball;

ALTER TABLE temp2 table RENAME TO ball_by_ball;

5. We also tried to normalize the data by creating two more table named *ground* and *match_by_team* as data has high reluctancies. So by creating two more tables we normalized it.For eg. We created table ground as follows:

CREATE TABLE Ground AS select Distinct(venue_name),City_Name,Country_Name from match order by country_name,City_name;

2.3 Data Statistics

Table	No. of tuples	time to load	size of raw dataset	size after cleanup
Team	13	179.608ms	343B	343B
Player match	13993	2342.310 ms	2.6 MB	904kB
Player	497	84.537 ms	34.6 kB	34.6kB
ball by ball	150451	8124.430 ms	24.5 MB	16MB
match	637	486.327 ms	113.4 kB	97KB
Ground	39	122.202 ms	-	16KB
match by team	150451	805.057 ms	-	6512KB

3 Section 3

3.1 User's view

- **Navigation bar** This is the complete map of our website. All pages can be viewed directly from here. Home, teams, venue and match are various categories in which the user can be interested independently. There is also a search bar that helps find the player's name with a prediction by the player's name so that the user does not need to type the entire or correct name. They can only type a later start and after that pattern the user will be able to see the list. The output will give all the information for the particular player from 2008 to 2017.

- **Home** This is the main page of the website. All player information can be viewed directly from here. Can be viewed according to the particular country, special season and special team. Statistics such as the highest six, four or wicket for a particular season can also be seen. Our interface is informative. Instead of comparing. So we mostly focused on the player statistics rather than the players.

- **Teams** The team page is helpful for giving users specific interfaces for team data. Only team data can be viewed from here such as how many matches have been played between 2008 and 2017. In this, how many won and how many lost. With a list of all the player parts of this team so far

- **Stadium** This page contains only the locations used to play matches from 2008 to 2017. Location with all information user might take interest like how many matches happened there and In how many team which bat first won the match. percentage winning of team which bat first.

- **Matches** This is the most important part of our website, which has complete information about all the matches between 2008 and 2017.User can search here like home page means for particular year, particular team or particular stadium. All entries also include which team won the match and what was the scoreboard for that match. The main entries of each match also contain information about where the match took place and who was the man of the match. Along with each scorecard, the scorecard of the player who bowls or bats in any innings also includes the entire team of each team at the bottom of the page. The scorecard consists of batsman, strike rate and bowler economy as well as runs, wickets, 4s, 6s. It also contains basic information about which team has won the data and margin, who has won the toss and what has been decided.

3.2 Functionalities

Functionalities used in our project are-

SUBQUERIES - We have used subqueries in some of our queries like -

select a1.sum,player_name from (select sum(runs_scored),striker from runs_in_ball where season_year = '2010' group by striker order by sum(runs_scored) desc) as a1 , player where player.Player_Id = a1.striker group by a1.striker,a1.sum,player_name order by a1.sum desc;

This query will give list of players with their runs in year 2010 in ipl in Descreasing order of their runs.i.e. Highest scorer of ipl of 2010 would be first tuple of output.

INDEXES - Indexes are used so it would be faster to scan data in the database like when we use where clause we need to go through database so if we use indexes it would be very easy to scan it. Postgres automatically build index on primary key of the table. We created index for newly created table Ground as -

CREATE INDEX ground_index ON Ground(Ground Name);

INTERSECT - We have used intersect in some queries to get column which are available in both queries like -

select distinct player.player_id , player.player_name, player.dob , player.batting_hand, player.bowling_skill , player.country_name from player where country_name = 'India' intersect select distinct player.player_id , player.player_name, player.dob , player.batting_hand, player.bowling_skill , player.country_name from player_match,player,mat where match.match_id = player_match.match_id and player_match.player_id = player.player_id and match.season_year = '2010';

Above query will give details of player who has played in ipl of 2010 and is from India.

VIEWS - We have used views in some of the queries otherwise queries would get long,complicated and slow.So, to optimize our query we used views. We have dropped all the queries which are used to show data only like -

create view total_matches as select match.season_year as season_year, count(player_match.player_team)/11 as total_matches from player_match,match where match.match_id = player_match.match_id AND player_match.player_team = 'Mumbai Indians' group by match.season_year order by match.season_year;

create view winning_matches as select match.season_year as season_year, count(player_match.player_team)/11 as winning_matches from player_match,match where match.match_id = player_match.match_id AND player_match.player_team = 'Mumbai Indians' AND match.match_winner = 'Mumbai Indians' group by match.season_year order by match.season_year;

select total_matches.season_year, total_matches.total_matches, winning_matches.winning_matches from total_matches,winning_matches where total_matches.season_year = winning_matches.season_year;

drop view total matches;

drop view winning matches;

LEFT OUTER JOIN - We have used left outer join to merge our two separate tables, one is calculated using query with existed table player in database on player_id to get all information about single player in a specific match to answer bowling or batting stats of that match for all player who played that match. query like -

*SELECT * FROM (SELECT * FROM (SELECT bowler, ROUND(SUM(CASE extra_type WHEN 'Wides' THEN 0 ELSE (CASE extra_type WHEN 'wides' THEN 0 ELSE (CASE extra_type WHEN 'Noballs' THEN 0 ELSE (CASE extra_type WHEN 'noballs' THEN 0 ELSE 1 END)END)END)END):: DECIMAL/6,1) AS over, SUM(bowler_wicket) AS wickets, SUM(runs_scored) AS run, SUM(extra_runs) AS extra_runs, SUM(CASE extra_type WHEN 'byes' THEN extra_runs ELSE (CASE extra_type WHEN 'Byes' THEN extra_runs ELSE (CASE extra_type WHEN 'legbyes' THEN extra_runs ELSE (CASE extra_type WHEN 'Legbyes' THEN extra_runs ELSE(CASE extra_type WHEN 'panalty' THEN extra_runs ELSE 0 END)END)END)END)END) AS bp_extra, SUM(CASE extra_type WHEN 'wides' THEN 1 ELSE (CASE extra_type WHEN 'Wides' THEN 1 ELSE 0 END) END) AS wides, SUM(CASE extra_type WHEN 'Noballs' THEN 1 ELSE 0 END) AS nb , ROUND(SUM(runs_scored+extra_runs) :: DECIMAL / ROUND(SUM (COALESCE(CASE extra_type WHEN 'No Extras' THEN 1 ELSE NULL END,CASE extra_type WHEN 'legbyes' THEN 1 ELSE 0 END))::DECIMAL /6,1) , 2) AS eco FROM ball_by_ball WHERE match_id = '335987' AND innings_no = '1' GROUP BY bowler) AS foo LEFT OUTER JOIN player ON player.player_id = foo.bowler;*

INNER JOIN - We have used inner join to join player table with computed batsman data table for

particular match. anyway data is complete and there is not any problem with player id in computed table and player name in player table. but still using inner-join would be a better option so that in case problem occur we can still get the valid information about some player-

*SELECT * FROM player INNER JOIN (SELECT striker,SUM(runs_scored) AS run, SUM(CASE extra_type WHEN 'Wides' THEN 0 ELSE (CASE extra_type WHEN 'wides' THEN 0 ELSE (CASE extra_type WHEN 'Noballs' THEN 0 ELSE (CASE extra_type WHEN 'noballs' THEN 0 ELSE 1 END)END)END)END) AS balls, SUM(CASE runs_scored WHEN 4 THEN 1 ELSE 0 END) AS fours, SUM(CASE runs_scored WHEN 6 THEN 1 ELSE 0 END) AS sixes , round(SUM(runs_scored) :: DECIMAL / count(CONCAT(over_id,ball_id,innings_no)) , 2)*100 AS sr, SUM(extra_runs) AS extra, count(player_out) AS outCount FROM ball_by_ball WHERE match_id = '335987' AND innings_no ='1' GROUP BY striker) AS foo ON foo.striker = player.player_id;*

Note : We have also used **COALESCE** in above queries for multiple **CASE STATEMENT** for finding conditional sum of same column to get entries regarding wide ball, dot ball, no ball etc. for scorecard of particular match.

3.3 Queries

1. **Batting stats**

CREATE view runs_in_ball AS SELECT runs_scored ,striker,season_year,match.match_id FROM ball_by_ball,match WHERE match.match_id= ball_by_ball.match_id; CREATE view v1 AS SELECT sum(runs_scored) AS runs_scored,player_name,striker,season_year FROM runs_in_ball,player WHERE striker=player.player_id AND player.player_name ='V Kohli' GROUP BY striker,player_name,season_year order by season_year desc; CREATE view v2 AS SELECT count(player_match.match_id),match.season_year FROM player_match,match,player WHERE player_match.match_id = match.match_id AND player.player_name = 'V Kohli' AND player.player_id=player_match.player_id GROUP BY match.season_year,player.player_id;CREATE table v4 AS SELECT coalesce(count(a1.sum),0),a1.season_year FROM (SELECT sum(runs_scored),striker,season_year FROM runs_in_ball,player WHERE striker=player.player_id AND player_name = 'V Kohli' GROUP BY striker,match_id,season_year order by sum(runs_scored) desc) AS a1 WHERE a1.sum ¿= 100 GROUP BY a1.season_year;INSERT into v4 values(0,2008);INSERT into v4 values(0,2009);INSERT into v4 values(0,2010);INSERT into v4 values(0,2011);INSERT into v4 values(0,2012);INSERT into v4 values(0,2013);INSERT into v4 values(0,2014);INSERT into v4 values(0,2015);INSERT into v4 values(0,2016);INSERT into v4 values(0,2017);CREATE view v40 AS SELECT max(coalesce),season_year FROM v4 GROUP BY season_year; CREATE table v3 AS SELECT coalesce(count(a1.sum),0),a1.season_year FROM (SELECT sum(runs_scored),striker,season_year FROM runs_in_ball,player WHERE striker=player.player_id AND player_name = 'V Kohli' GROUP BY striker,match_id,season_year order by sum(runs_scored) desc) AS a1 WHERE a1.sum ¿= 50 AND a1.sum ¡100 GROUP BY a1.season_year; INSERT into v3 values(0,2008);INSERT into v3 values(0,2009);INSERT into v3 values(0,2010);INSERT into v3 values(0,2011);INSERT into v3 values(0,2012);INSERT into v3 values(0,2013);INSERT into v3 values(0,2014);INSERT into v3 values(0,2015);INSERT into v3 values(0,2016);INSERT into v3 values(0,2017);CREATE view v30 AS SELECT max(coalesce),season_year FROM v3 GROUP BY season_year; CREATE view v5 AS SELECT max(a1.sum),a1.season_year FROM (SELECT sum(runs_scored),striker ,season_year FROM runs_in_ball,player WHERE striker=player.player_id AND player_name = 'V Kohli' GROUP BY striker,match_id,season_year order by sum(runs_scored) desc) AS a1 GROUP BY a1.season_year; SELECT v1.runs_scored,v5.max AS HS, v2.count AS matches , v30.max AS fifties , v40.max AS centuries,v1.season_year AS season FROM v1,v2,v3,v4,v5,v30,v40 WHERE v1.season_year = v2.season_year AND v30.season_year = v2.season_year AND v30.season_year = v40.season_year AND v2.season_year = v3.season_year AND v3.season_year = v4.season_year AND v4.season_year = v5.season_year GROUP BY v1.runs_scored , HS , matches , fifties,centuries,season order by season desc; drop view v1;drop view v2;drop view v5;drop view v30;drop view v40;drop view runs_in_ball;drop table v3;drop table v4;

2. **Bowling Stats**

create TABLE v2 as select match.season_year,count(ball_by_ball.bowler) from match,ball_by_ball,player WHERE ball_by_ball.bowler_wicket = '1' AND player.player_name = 'B Kumar' AND ball_by_ball.bowler = player.player_id AND match.match_id = ball_by_ball.match_id group by match.season_year,ball_by_ball.bowler; INSERT into v2 values(2008,0);_INSERT into v2 values(2009,0);INSERT into v2 values(2010,0);INSERT

into v2 values(2011,0);_INSERT into v2 values(2012,0); INSERT into v2 values(2013,0); INSERT into v2 values(2014,0);_ INSERT into v2 values(2015,0);INSERT into v2 values(2016,0);INSERT into v2 values(2017,0);CREATE view wickets _AS SELECT season_year,max(COUNT) as count FROM v2 GROUP BY season_year; create view total_matches as select_ match.season_year as season_year, count(player_match.player_id) as COUNT from match, player_match,player_where player.player_id = player_match.player_id AND player_match.match_id = match.match_id AND player.player_name = 'B Kumar' group by match.season_year,player_match.player_id;create table v3 AS select match.season_year as season_year,_ count(ball_by_ball.bowler) as COUNT from match,ball_by_ball,player where player.player_id = ball_by_ball.bowler AND_ ball_by_ball.match_id = match.match_id AND player.player_name = 'B Kumar'group by match.season_year,ball_by_ball.bowler ;INSERT into v3 values(2008,0);INSERT into v3 values(2009,0);INSERT into v3 values(2010,0);INSERT into v3 values(2011,0);INSERT into v3 values(2012 ,0);INSERT into v3 values(2013,0);INSERT into v3 values(2014,0);INSERT into v3 values(2015,0);INSERT into v3 values(2016,0);INSERT into v3 values(2017,0);CREATE view totalballs AS SELECT season_year,max(COUNT) as count FROM v3 GROUP BY season_year;create TABLE v4 as select match.season_year, count(ball_by_ball.bowler) from match,ball_by_ball,player WHERE ball_by_ball.extra_type = 'wides' AND player.player_name = 'B Kumar' AND ball_by_ball.bowler = player.player_id AND match.match_id = ball_by_ball.match_id group by match.season_year,ball_by_ball.bowler;INSERT into v4 values(2008,0);INSERT into v4 values(2009,0);INSERT into v4 values(2010,0);INSERT into v4 values(2011,0);INSERT into v4 values(2012 ,0);INSERT into v4 values(2013,0);INSERT into v4 values(2014,0);INSERT into v4 values(2015,0);INSERT into v4 values(2016,0);INSERT into v4 values(2017,0);CREATE view totalwides AS SELECT season_year,max(COUNT) as count FROM v4 GROUP BY season_year; create TABLE v5 as select match.season_year, count(ball_by_ball.bowler) from match,ball_by_ball,player WHERE ball_by_ball.extra_type = 'Noballs' AND player.player_name = 'B Kumar' AND ball_by_ball.bowler = player.player_id AND match.match_id = ball_by_ball.match_id group by match.season_year, ball_by_ball.bowler ;INSERT into v5 values(2008,0);INSERT into v5 values(2009,0);INSERT into v5 values(2010,0); INSERT into v5 values(2011,0);INSERT into v5 values(2012 ,0); INSERT into v5 values(2013,0);INSERT into v5 values(2014,0);INSERT into v5 values(2015,0);INSERT into v5 values(2016,0);INSERT into v5 values(2017,0);CREATE view totalnoballs AS SELECT season_year,max(COUNT) as count FROM v5 GROUP BY season_year; select total_matches.season_year as season_year, total_matches.count as total_matches, totalballs.count as totalballs, wickets.count as wickets,totalwides.count as totalwides, totalnoballs.count as totalnoballs from total_matches,totalballs,totalwides,totalnoballs,wickets where total_matches.season_year = wickets.season_year AND wickets.season_year = totalballs.season_year AND totalwides.season_year = wickets.season_year AND totalwides.season_year = totalnoballs.season_year order by season_year;drop view wickets;drop view total_matches;drop view totalballs;drop view totalnoballs;drop view totalwides; drop table v3; drop table v4;drop table v5;

3. **Venue stats**

create view total_venue_matches as SELECT match.venue_name as venue_name, ground.city_name as city_name,ground.country_name as country_name,count(match.venue_name) as total_venue_matches from ground,match where ground.venue_name = match.venue_name group by match.venue_name,ground.city_name,ground.country_name;

create view winning_firstinnings_matches as select match.venue_name as venue_name, ground.city_name as city_name,ground.country_name as country_name,count(match.venue_name) as winning_firstinnings_matches from ground,match where ground.venue_name = match.venue_name AND match.win_type = 'wickets' group by match.venue_name,ground.city_name,ground.country_name;

select total_venue_matches.venue_name, total_venue_matches.city_name, total_venue_matches.country_name, total_venue_matches.total_venue_matches, winning_firstinnings_matches.winning_firstinnings_matches, (100 winning_firstinnings_matches.winning_firstinnings_matches/total_venue_matches.total_venue_matches) as perc_battingfirst from total_venue_matches, winning_firstinnings_matches where total_venue_matches.venue_name = winning_firstinnings_matche*

drop view total_venue_matches; drop view winning_firstinnings_matches;

4. **Highest sixes in any year**

select a1.a2 as sixes,player.player_name from (select striker,count(striker) as a2 from ball_by_ball,match where runs_scored = '6' and match.season_year = '2010' and match.match_id = ball_by_ball.match_id group by striker order by count(striker) desc)as a1 , player where player.player_id = a1.striker group by a1.a2 , player.player_name order by a1.a2 desc;

5. **Highest four in any year**

select a1.a2 as fours,player.player_name from (select striker,count(striker) as a2 from ball_by_ball,match where runs_scored = '4' and match.season_year = '2010' and match.match_id = ball_by_ball.match_id group by striker order by count(striker) desc)as a1 , player where player.player_id = a1.striker group by a1.a2 , player.player_name order by a1.a2 desc;

6. **Highest Wicket taker**

select a1.a2 as wickets,player.player_name from (select bowler,count(bowler) as a2 from ball_by_ball,match where bowler_wicket = '1' and match.season_year = '2010' and match.match_id = ball_by_ball.match_id group by bowler order by count(bowler) desc)as a1 , player where player.player_id = a1.bowler group by a1.a2 , player.player_name order by a1.a2 desc;

7. **Team stats**

*select a1.season_year,a1.total_matches,a2.winning_matches, 100*a2.winning_matches/a1.total_matches as winning_percent from (select match.season_year as season_year, count(player_match.player_team)/11 as total_matches from player_match,match where match.match_id = player_match.match_id AND player_match.player_team = 'Mumbai Indians' group by match.season_year order by match.season_year) as a1, (select match.season_year as season_year, count(player_match.player_team)/11 as winning_matches from player_match,match where match.match_id = player_match.match_id AND player_match.player_team = 'Mumbai Indians' AND match.match_winner = 'Mumbai Indians' group by match.season_year order by match.season_year) as a2 where a1.season_year = a2.season_year;*

8. **Player of some country playing in some year**

select distinct player.player_id , player.player_name, player.dob , player.batting_hand, player.bowling_skill , player.country_name from player where country_name = 'India' intersect select distinct player.player_id , player.player_name, player.dob , player.batting_hand, player.bowling_skill , player.country_name from player_match,player,mat where match.match_id = player_match.match_id and player_match.player_id = player.player_id and match.season_year = '2010';

9. **Particular Match, bowling Scorecard**

*SELECT * FROM (SELECT * FROM (SELECT bowler, ROUND(SUM(CASE extra_type WHEN 'Wides' THEN 0 ELSE (CASE extra_type WHEN 'wides' THEN 0 ELSE (CASE extra_type WHEN 'Noballs' THEN 0 ELSE (CASE extra_type WHEN 'noballs' THEN 0 ELSE 1 END)END)END)END)END):: DECIMAL/6,1) AS over, SUM(bowler_wicket) AS wickets, SUM(runs_scored) AS run, SUM(extra_runs) AS extra_runs, SUM(CASE extra_type WHEN 'byes' THEN extra_runs ELSE (CASE extra_type WHEN 'Byes' THEN extra_runs ELSE (CASE extra_type WHEN 'legbyes' THEN extra_runs ELSE (CASE extra_type WHEN 'Legbyes' THEN extra_runs ELSE(CASE extra_type WHEN 'panalty' THEN extra_runs ELSE 0 END)END)END)END)END) AS bp_extra, SUM(CASE extra_type WHEN 'wides' THEN 1 ELSE (CASE extra_type WHEN 'Wides' THEN 1 ELSE 0 END) END) AS wides, SUM(CASE extra_type WHEN 'Noballs' THEN 1 ELSE 0 END) AS nb , ROUND(SUM(runs_scored+extra_runs) :: DECIMAL / ROUND(SUM (COALESCE(CASE extra_type WHEN 'No Extras' THEN 1 ELSE NULL END,CASE extra_type WHEN 'legbyes' THEN 1 ELSE 0 END))::DECIMAL /6,1) , 2) AS eco FROM ball_by_ball WHERE match_id = '335987' AND innings_no = '1' GROUP BY bowler) AS foo LEFT OUTER JOIN player ON player.player_id = foo.bowler;*

10. **Particular Match, batting Scorecard**

*SELECT * FROM player INNER JOIN (SELECT striker,SUM(runs_scored) AS run, SUM(CASE extra_type WHEN 'Wides' THEN 0 ELSE (CASE extra_type WHEN 'wides' THEN 0 ELSE (CASE extra_type WHEN 'Noballs' THEN 0 ELSE (CASE extra_type WHEN 'noballs' THEN 0 ELSE 1 END)END)END)END) AS balls, SUM(CASE runs_scored WHEN 4 THEN 1 ELSE 0 END) AS fours, SUM(CASE runs_scored WHEN 6 THEN 1 ELSE 0 END) AS sixes , round(SUM(runs_scored) :: DECIMAL / count(CONCAT(over_id,ball_id,innings_no)) , 2)*100 AS sr, SUM(extra_runs) AS extra, count(player_out) AS outCount FROM ball_by_ball WHERE match_id = '335987' AND innings_no ='1' GROUP BY striker) AS foo ON foo.striker = player.player_id;*

3.4 Query Time

Query No.	Time
1	1008.689 ms
2	389.483 ms
3	125.936 ms
4	52.273 ms
5	51.794 ms
6	55.572 ms
7	5.855 ms
8	22.071 ms
9	2731.246 ms
10	207.998 ms