Team SKY – IPL Analysis Database

Project Report

Manikanta Kalyan
Shylesh Kadam
Yaswanth Reddy Thippaluri
School of Engineering and Applied Sciences
University at Buffalo

***Abstract*— In order to determine individual performance of a player, which player can be picked by the teams in the forthcoming seasons, we will look for observations such as each player's stats, the contribution of each player for his team and other metrics. These observations will help the cricket analysts to make healthy decisions.**

Database VS Excel :

To attain these kinds of results we use databases instead of excel because of the following reasons:

- Excel is significantly slower than database systems. Sheets and Excel are excellent for charting, presentations, and quick analysis, but they may fall short for jobs that demand for heavier lifting.

- Most of those tasks can be completed more quickly with DB systems, and the entire process is more user-friendly.The Database systems also separate analysis from data.

- Moreover,DB systems provide better Data Integrity,consistency and can handle large datasets with faster manipulation.

I. TARGET USER

Users :

The IPL database can be used by various users like :

- Cricket analysts
- Cricket Coaches
- Franchise owners
- Various Cricketing websites and applications like Cricbuzz, Cricinfo.com by ESPN.
- Data Analysts who want to predict the player or team performances in the upcoming seasons based on the previous years stats.

Administrators:

Administrators can be of various types

- For applications and websites related to Cricketing news they are maintained by private firms.

Real-life scenario :

In Real life scenarios Cricket analysts, Coaches and Franchise owners use the data to understand and summarize the batting/bowling/fielding performances. For example below are some of the metrics we can get by designing a database for IPL data.

Batting metrics

- Highest run scorer for the entire season.
- Strike rate of individual players.
- Most boundaries hitting players in a season.
- Count of centuries of a individual player,
- Runs scored in powerplay by an individual player.

Bowling metrics

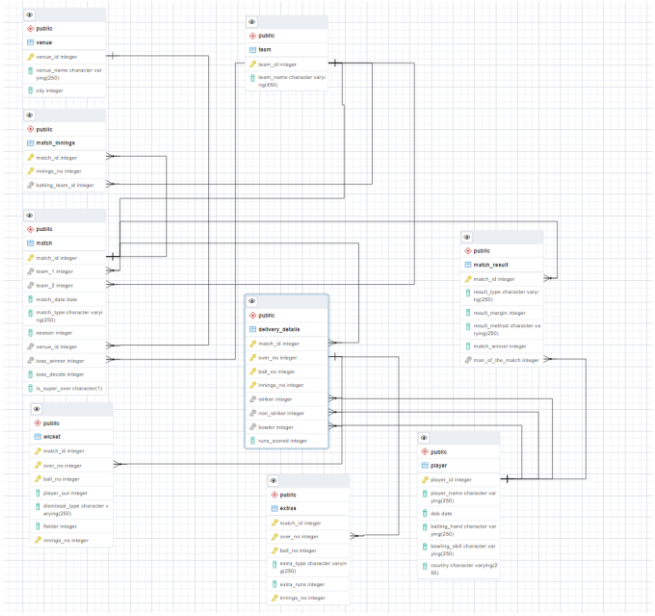
- Highest wickets taken player for the entire season.
- Economy.
- Death over and power play economies of an individual player.

Fielding metrics

- Most number of catches caught by a player for the entire season.

By summarizing this kind of data from all the IPL seasons, the users will be able to make some crucial decisions for the upcoming matches. And also cricketing news related websites or applications use these databases in their applications to continuously store/update the new incoming data and to calculate the updated player and team statistics.

II. ER DIAGRAM



A. *Database implementation*

Brief description of each relation

Player:
This relation shows details about each player.

Venue:
This relation shows the venue details(place where the particular match is played)

Team:
This relation shows the team details like each team's ID and name

Match:
This relation contains the details of a particular match being played.

Match_Innings:
This relation shows the innings details of each match played.

Match_Result:
This relation shows the results of the particular match along with the result margin and key player of the match.

Delivery_Details:
This is the main relation in our database which contains details of each ball being delivered.

Extras:
This relation shows the extras(runs that are not scored by a batsmen) in the particular match.

Wicket:
This relation shows the wicket details of a particular player like the way of his dismissal type.

B. *Attributes*

Player

Attributes	Data Type	Description
Player_id	Integer	Values to uniquely identify all the rows of the Player table. Value cannot be NULL.
Player_Name	Varchar	Names of the all the players and the player name cannot be NULL
DOB	Date	Date of birth of all players. Date of Birth cannot be NULL.
Batting_hand	Varchar	Data indicating ‘right’ hand or ‘left’ hand batsmen. Batting_hand cannot be NULL.
Bowling_skill	Varchar	Bowling style of a Bowler like leg-spinner, off-spinner etc. Bowling_skill can be NULL
Country	Varchar	Data related to the player's country and cannot be NULL

Venue

Attributes	Data Type	Description
Venue_Id	Integer	Values to uniquely identify all the available venues. These values cannot be NULL.
Venue_Name	Varchar	Name of the venue. Venue name cannot be NULL.
City	Varchar	Names of the cities that the venues are in. City names cannot be NULL.

Team

Attributes	Data Type	Description
Team_Id	Integer	Values to uniquely identify all the teams in ipl. These values cannot be NULL.
Team_Name	Varchar	Names of teams. Team names cannot be NULL.

Match

Attributes	Data Type	Description
Match_Id	Integer	Values to uniquely identify all the matches. Match IDs cannot be NULL.
Team_1	Integer	Team id’s of one of the teams that is participating

		in a particular match. Team_1 cannot be NULL.
Team_2	Integer	Team id's of the other team that is participating in a particular match Team_2 cannot be NULL.
Match_Date	Date	Date of the match. Match_date cannot be NULL.
Match_Type	Varchar	Type of the match like Qualifier, eliminator, Final etc. Match_type cannot be NULL.
Venue_Id	Integer	Representing the venue of the match that took place. Venue id cannot be NULL.
Toss_Winner	Integer	Team id values who won the toss. Toss winner cannot be NULL.
Toss_Decision	Varchar	Team's choice to 'bat' or 'bowl' after winning the toss. Toss_Decision cannot be NULL.
Is_Super_Over	Char	the character 'Y' or 'N' representing a Super over took place or not for a match. Is_Super_Over cannot be NULL.

Match_Innings

Attributes	Data Type	Description
Match_Id	Integer	Values to uniquely identify all the matches. Match IDs cannot be NULL.
Innings_No	Integer	Values represent '1' first innings, '2' second innings. '3' and '4' in case of super overs. Innings_No cannot be NULL
Batting_Team_Id	Integer	Team id value who chooses to bat in that innings. Batting team id cannot be NULL.

Match_Result

Attributes	Data Type	Description
Match_Id	Integer	Values to uniquely identify all the matches. Match IDs cannot be NULL.
Result_Type	Varchar	Represents Won by 'runs' or Won by 'wickets'. Result Type
		cannot be NULL.Innings_No cannot be NULL
Result_Margin	Integer	No.of runs or wickets by which the team won. Result margin can be NULL in case of no result.
Result_Method	Varchar	Method by which the result of the match has been decided. For special cases or we put NA ex: duckworth-lewis (DLS). Result method can be NULL.
Match_Winner	Integer	Team id of the team who won. Match_winner can have NULLvalues in case the match results in a draw.
Man_of_the_Match	Integer	Player id who won Man of the match. It can be NULL.

Delivery_Details

Attributes	Data Type	Description
Match_Id	Integer	Values to uniquely identify all the matches. Match IDs cannot be NULL.
Over_No	Integer	Values representing number of the over and cannot be NULL
Ball_No	Integer	Values representing the ball number in particular over and cannot be NULL
Innings_No	Integer	Values represent '1' first innings, '2' second innings. '3' and '4' in case of super overs. Innings_No cannot be NULL
Striker	Integer	Player_id who is on strike and cannot be NULL
Non_Striker	Integer	Player_id who is not on strike and cannot be NULL
Runs_Scored	Integer	Number of runs scored by the striker for that particular delivery and cannot be NULL
Bowler	Integer	Played_id of the bowler and cannot be NULL

Extras

Attributes	Data Type	Description
Match_Id	Integer	Values to uniquely identify all the matches. Match IDs cannot be NULL.
Over_No	Integer	Values representing number of the over and cannot be NULL
Ball_No	Integer	Values representing the ball number in particular over and cannot be NULL
Extra_Type	Varchar	Indicates extra type(wide,no ball) and can be NULL
Extra_Runs	Integer	Number of runs from extras and cannot be NULL
Innings_No	Integer	Values represent '1' first innings, '2' second innings. '3' and '4' in case of super overs. Innings_No cannot be NULL

Wicket

Attributes	Data Type	Description
Match_Id	Integer	Values to uniquely identify all the matches. Match IDs cannot be NULL.
Over_No	Integer	Values representing number of the over and cannot be NULL
Ball_No	Integer	Values representing the ball number in particular over and cannot be NULL
Player_Out	Integer	Played_id of the player who got out and cannot be NULL
Dismissal_Type	Varchar	Type of dismissal(caught,runout,bowled) and cannot be NULL
Fielder	Integer	Player_id of player who is responsible for wicket and can be NULL
Innings_No	Integer	Values represent '1' first innings, '2' second innings. '3' and '4' in case of super overs. Innings_No cannot be NULL
Bowler	Integer	Played_id of the bowler and cannot be NULL

Primary and Foreign keys:

Table Name	Primary Key	Foreign Keys - Referenced table
Player	Player_Id	-
Venue	Venue _Id	-
Team	Team_Id	-
Match	Match_ID	Team_1 - Team Team_2 - Team Toss_Winner - Team Venue_Id - Venue
Match_Innings	Innings_No, Match_Id	Match_Id - Match Batting_team_id – Team
Match_Result	Match_Id	Match_Id - Match Man_of_the_Match - Player
Delivery Details	Match_Id, Over_No, Ball_No, Innings_No	Match_Id - Match Striker - Player Non_Striker - Player Bowler - Player
Extras	Match_Id, Over_No, Ball_No, Innings_No	(Match_Id ,Over_No, Ball_No, Innings_No) - Delivery_Details
Wicket	Match_Id, Over_No, Ball_No, Innings_No	(Match_Id, Over_No, Ball_No, Innings_No) - Delivery_Detais (Bowler,Fielder,Player_Out) - Player

Note:
Since we are maintaining Historical data in the database, we are not allowed to delete any parent/primary key data. So, we are not mentioning any actions to be taken on foreign key references whenever primary key data is deleted as the default behavior of the database server prevents DELETE and MERGE statements from deleting data in a table that another table references within a primary-key foreign-key relationship.

III. FUNCTIONAL DEPENDENCIES TO SHOW THAT TABLES ARE IN BCNF

Player Table
FD:
Player_Id → Player_Name, DOB, Batting_hand,Bowling_skill, Country

{Player_Id} + = {Player_Id, Player_Name, DOB, Batting_hand,Bowling_skill, Country}

The closure of Player_Id is the set of all attributes in the Player Table. So, Player_Id is the super key.
So, the above FD is in BCNF.

Venue Table
FD:
Venue_Id → Venue_Name, City

{Venue_Id }+ = {Venue_Id, Venue_Name, City}

The closure of Venue_Id is the set of all attributes in the Venue Table. So, Venue_Id is the super key.
So, the above FD is in BCNF.

Team Table
FD:
Team_Id → Team_Name

{Team_Id}+ = {Team_Id, Team_Name}

The closure of Team_Id is the set of all attributes in the Team Table. So, Team_Id is the super key.
So, the above FD is in BCNF.

Match Table
FD:
Match_Id → Team_1, Team_2, Match_Date, Match_Type, Venue_Id , Toss_Winner, Toss_Decision, Is_Super_Over

{Match_Id}+ = {Match_Id, Team_1, Team_2, Match_Date, Match_Type, Venue_Id , Toss_Winner, Toss_Decision, Is_Super_Over}

The closure of Match_Id is the set of all attributes in the Match Table. So, Match_Id is the super key.
So, the above FD is in BCNF.

Match_Innings Table
FD:
Match_Id, Innings_No → Batting_Team_Id

{Match_Id, Innings_No}+ = { Match_Id, Innings_No, Batting_Team_Id}

The closure of Match_Id, Innings_No is the set of all attributes in the Match_Innings Table. So, Match_Id, Innings_No is the super key. So, the above FD is in BCNF.

Match_Result Table
FD:
Match_Id → Result_Type, Result_Margin, Result_Method, Match_Winner, Man_of_the_Match

{Match_Id}+ = {Match_Id, Result_Type, Result_Margin, Result_Method, Match_Winner, Man_of_the_Match}

The closure of Match_Id is the set of all attributes in the Match_Result Table. So, Match_Id is the super key. So, the above FD is in BCNF.

Delivery_Details Table
FD:
Match_Id, Over_No, Ball_No, Innings_No → Striker, Non_Striker, Runs_Scored, Bowler

{Match_Id, Over_No, Ball_No, Innings_No}+ = {Match_Id, Over_No, Ball_No, Innings_No, Striker, Non_Striker, Runs_Scored, Bowler}

The closure of "Match_Id, Over_No, Ball_No, Innings_No" is the set of all attributes in the Delivery_Details Table. So "Match_Id, Over_No, Ball_No, Innings_No" is the super key. So, the above FD is in BCNF.

Extras Table
FD:
Match_Id, Over_No, Ball_No, Innings_No → Extra_Type, Extra_Runs

{Match_Id, Over_No, Ball_No, Innings_No}+ = {Match_Id, Over_No, Ball_No, Innings_No, Extra_Type, Extra_Runs}

The closure of "Match_Id, Over_No, Ball_No, Innings_No" is the set of all attributes in the Extras Table. So "Match_Id, Over_No, Ball_No, Innings_No" is the super key. So, the above FD is in BCNF.

Wicket Table
FD:
Match_Id, Over_No, Ball_No, Innings_No → Player_Out, Bowler, Dismissal_Type, Fielder

{Match_Id, Over_No, Ball_No, Innings_No}+ = { Match_Id, Over_No, Ball_No, Innings_No, Player_Out, Bowler, Dismissal_Type, Fielder}

The closure of "Match_Id, Over_No, Ball_No, Innings_No" is the set of all attributes in the wicket Table. So "Match_Id, Over_No, Ball_No, Innings_No" is the super key. So, the above FD is in BCNF

IV. DATA SOURCE AND NOVELITY

a)Data Source :

Dataset : https://www.kaggle.com/datasets/vora1011/ipl-2008-to-2021-all-match-dataset
Dataset 2022: https://www.kaggle.com/datasets/vora1011/ipl-2022-player-statistic

b) Novelty :
The first Datasource had data until year 2021 and second datasource has Data for year 2022.Both sources had only two csv files with all the information in it.We extracted each table(as per our schema design) from the above sources.
Problems faced and improvements :

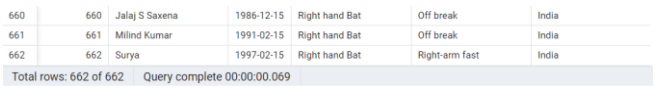
- While extracting each table from the data source ,we had to clean data and transform it accordingly as per our design.
- We improved database design by adding constraints and removing redundancy ,so that the tables are in best form.

- Both the data sources had discrepancy in data(Some of the player information was missing), We identified the missing data using Vlook up excel commands, and then we manually inserted each player's information by using online resources.

V. SQL QUERIES

1.Insert Query
INSERT INTO player VALUES (662,'Surya','1997-02-15','Right hand Bat','Right-arm fast','India');

Result:



2.Update Query
UPDATE player SET dob = '1997-03-20' WHERE player_name='Surya';

Result :

3. Delete query
DELETE FROM player WHERE player_name = 'Surya';

Result:

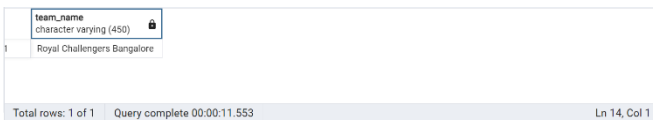
4. Find the youngest player in the IPL.
select t.player_name from (select rank() over(order by dob desc) as srank,player_name from player) as t where t.srank = 1;

Result:

player_name character varying (250)	dob date
D Brevis	2003-04-29

tal rows: 1 of 1 Query complete 00:00:00.082

5. Find the team that has won the most number of times in 'M Chinnaswamy Stadium' in the entire IPL.
select team_name from team where team_id in (select match_winner from match_result where match_id in(select match_id from match where venue_id in (select venue_id from venue where venue_name = 'M Chinnaswamy Stadium')) group by match_winner order by count(*) desc limit 1);
Result:



6. Find the top 5 players who have taken the most number of catches in the IPL.
select player_name ,count(*) as number_of_catches from wicket inner join player on player.player_id = wicket.fielder wheredismissal_type = 'caught' group by player_name order by number_of_catches desc limit 5;

Result:

	player_name character varying (250)	number_of_catches bigint
1	MS Dhoni	135
2	KD Karthik	133
3	AB de Villiers	120
4	SK Raina	106
5	KA Pollard	97

Total rows: 5 of 5 Query complete 00:00:00.050

7. List of top 5 players who bowled most dot balls.
select player.player_name,count(runs_scored) as DotBall from delivery_details inner join player on player.player_id = delivery_details.bowler where runs_scored = 0 group by player.player_name order by DotBall desc;

Result:

	player_name character varying (250)	dotball bigint
1	B Kumar	1581
2	R Ashwin	1548
3	SP Narine	1468
4	Harbhajan Singh	1390
5	SL Malinga	1357

Total rows: 5 of 5 Query complete 00:00:00.119

8. Find the top 5 teams who scored most number of fours and sixes in IPL

Select Team.Team_name, Sum(Case when delivery_details. runs_scored=4 Then 1 else 0 end) As Fours, Sum (Case when delivery_details.runs_scored = 6 Then 1 else 0 end) As Sixes from delivery_details Inner join Match on delivery_details. match_id = match.match_id Inner join team on match.team_1=Team.team_id group by Team.Team_name order by Fours desc ;

Result:

	team_name character varying (450)	fours bigint	sixes bigint
1	Royal Challengers Bangalo...	3212	1583
2	Mumbai Indians	3144	1298
3	Chennai Super Kings	2929	1305
4	Kolkata Knight Riders	2830	1161
5	Kings XI Punjab	2652	974

9.Find the venue name and city where the final took place throughout the years.

select distinct venue_name,city from venue where venue_id in (select venue_id from match where match_type = 'Final');

Result:

	venue_name character varying (250)	city character varying (250)
1	Dr DY Patil Sports Academy	Mumbai
2	Dubai International Cricket Stadi...	Abu Dhabi
3	Eden Gardens	Kolkata
4	M Chinnaswamy Stadium	Bangalore
5	MA Chidambaram Stadium	Chennai
6	Narendra Modi Stadium	Ahmedabad
7	New Wanderers Stadium	Johannesburg
8	Rajiv Gandhi International Stadium	Hyderabad
9	Wankhede Stadium	Mumbai

10. The top 5 individual highest scores and player names in IPL.

Select s.player_name, sum(d.runs_scored) as Runs_Scored, m.match_date as runs from delivery_details d inner join player s ON s.player_id = d.striker inner join match m on d.match_id=m.match_id group by s.player_id,d.match_id,m.match_date order by 2 desc limit 5

Result:

player_name character varying (250)	runs_scored bigint	runs date
CH Gayle	175	2013-04-23
BB McCullum	158	2008-04-18
Q de Kock	140	2022-05-18
AB de Villiers	133	2015-05-10
KL Rahul	132	2020-09-24

VI. QUERY ANALYSIS AND OPTIMIZATION

1. Find the player who scored more than 100 runs with only 6's in a single innings.

explain ANALYSE select s.player_name,m.match_date, sum(d.runs_scored) as runs from delivery_details d inner join player s ON s.player_id = d.striker inner join match m on d.match_id=m.match_id where d.runs_scored =6 group by s.player_id, m.match_id having sum (d.runs_scored) >=100 order by 2 desc;

Result:

	player_name character varying (250)	match_date date	runs bigint
1	CH Gayle	2013-04-23	102

Before indexing:

The query took 46.662 ms before applying Index on the table.We have then created an Index to reduce the execution time.

	QUERY PLAN text
10	-> Hash Join (cost=22.87..4536.76 rows=10394 width=22) (actual time=0.170..36.370 rows=10666 loops=1)
11	Hash Cond: (d.striker = s.player_id)
12	-> Seq Scan on delivery_details d (cost=0.00..4486.43 rows=10394 width=12) (actual time=0.009..33.221 rows=10666 loops=1)
13	Filter: (runs_scored = 6)
14	Rows Removed by Filter: 215288
15	-> Hash (cost=14.61..14.61 rows=661 width=14) (actual time=0.154..0.155 rows=662 loops=1)
16	Buckets: 1024 Batches: 1 Memory Usage: 40kB
17	-> Seq Scan on player s (cost=0.00..14.61 rows=661 width=14) (actual time=0.007..0.070 rows=662 loops=1)
18	-> Hash (cost=18.50..18.50 rows=950 width=8) (actual time=0.235..0.236 rows=950 loops=1)
19	Buckets: 1024 Batches: 1 Memory Usage: 46kB
20	-> Seq Scan on match m (cost=0.00..18.50 rows=950 width=8) (actual time=0.017..0.116 rows=950 loops=1)
21	Planning Time: 0.597 ms
22	Execution Time: 46.662 ms

Creating index:

create INDEX Runs on Delivery_details using hash(runs_scored);

After applying Index:

As we can see,after applying indexing on the table on runs_scored column the execution time has been reduced to 10.054 ms

-> Bitmap Heap Scan on delivery_details d (cost=319.33..2109.28 rows=10236 width=12) (actual time=0.556..3.994 rows=10666 loops=1)
Recheck Cond: (runs_scored = 6)
Heap Blocks: exact=1647
-> Bitmap Index Scan on runs (cost=0.00..316.77 rows=10236 width=0) (actual time=0.412..0.413 rows=10666 loops=1)
Index Cond: (runs_scored = 6)
-> Hash (cost=14.61..14.61 rows=661 width=14) (actual time=0.119..0.122 rows=661 loops=1)
Buckets: 1024 Batches: 1 Memory Usage: 40kB
-> Seq Scan on player s (cost=0.00..14.61 rows=661 width=14) (actual time=0.005..0.055 rows=661 loops=1)
-> Hash (cost=18.50..18.50 rows=950 width=8) (actual time=0.159..0.160 rows=950 loops=1)
Buckets: 1024 Batches: 1 Memory Usage: 46kB
-> Seq Scan on match m (cost=0.00..18.50 rows=950 width=8) (actual time=0.008..0.081 rows=950 loops=1)
Planning Time: 0.328 ms
Execution Time: 10.054 ms

2. Find the number of deliveries faced by a particular batsmen in a IPL

EXPLAIN ANALYSE select count(*) as Balls_Faced from delivery_details where striker in (select player_id from player where player_name ='S Dube');

Result:

	balls_faced bigint
1	541

Before Indexing:

The query took 51.196 ms before applying Index on the table.We have then created an Index to reduce the execution time.

	QUERY PLAN text
1	Finalize Aggregate (cost=4359.17..4359.18 rows=1 width=8) (actual time=46.926..51.164 rows=1 loops=1)
2	-> Gather (cost=4359.05..4359.16 rows=1 width=8) (actual time=46.275..51.156 rows=2 loops=1)
3	Workers Planned: 1
4	Workers Launched: 1
5	-> Partial Aggregate (cost=3359.05..3359.06 rows=1 width=8) (actual time=22.786..22.787 rows=1 loops=2)
6	-> Hash Join (cost=16.28..3358.55 rows=201 width=0) (actual time=0.399..22.763 rows=271 loops=2)
7	Hash Cond: (delivery_details.striker = player.player_id)
8	-> Parallel Seq Scan on delivery_details (cost=0.00..2991.14 rows=132914 width=4) (actual time=0.005..9.898 rows=112977 loops=2)
9	-> Hash (cost=16.26..16.26 rows=1 width=4) (actual time=0.109..0.110 rows=1 loops=1)
10	Buckets: 1024 Batches: 1 Memory Usage: 9kB
11	-> Seq Scan on player (cost=0.00..16.26 rows=1 width=4) (actual time=0.089..0.105 rows=1 loops=1)
12	Filter: ((player_name)::text = 'S Dube'::text)
13	Rows Removed by Filter: 661
14	Planning Time: 0.183 ms
15	Execution Time: 51.196 ms

Creating Index

create index idx_striker on delivery_details(striker);

After applying Index:

As we can see,after applying indexing on the table on striker column the execution time has been reduced to 0.302 ms

	QUERY PLAN text
1	Aggregate (cost=1003.92..1003.93 rows=1 width=8) (actual time=0.273..0.274 rows=1 loops=1)
2	-> Nested Loop (cost=11.69..1003.06 rows=342 width=0) (actual time=0.118..0.249 rows=541 loops=1)
3	-> Seq Scan on player (cost=0.00..16.26 rows=1 width=4) (actual time=0.051..0.060 rows=1 loops=1)
4	Filter: ((player_name)::text = 'S Dube'::text)
5	Rows Removed by Filter: 661
6	-> Bitmap Heap Scan on delivery_details (cost=11.69..982.58 rows=422 width=4) (actual time=0.064..0.152 rows=541 loops=1)
7	Recheck Cond: (striker = player.player_id)
8	Heap Blocks: exact=39
9	-> Bitmap Index Scan on idx_striker (cost=0.00..11.58 rows=422 width=0) (actual time=0.057..0.057 rows=541 loops=1)
10	Index Cond: (striker = player.player_id)
11	Planning Time: 1.989 ms
12	Execution Time: 0.302 ms

3. Top 5 scoring runs with other players when Virat Kohli is on non strike in individual matches.

EXPLAIN analyse select sum(runs_scored) as runs_scored from delivery_details where non_striker in (select player_id from player where player_name ='V Kohli') group by match_id, striker, non_striker order by 1 desc limit 5;

Result:

runs_scored bigint
133
129
127
101
83

Before Indexing:

Before indexing the query took 55.494 ms then we have created an index on non-striker

-> Parallel Seq Scan on delivery_details (cost=0.00..2991.14 rows=132914 width=16) (actual time=0.009..7.746 rows=112977 loops=
-> Hash (cost=16.26..16.26 rows=1 width=4) (actual time=0.333..0.334 rows=1 loops=2)
Buckets: 1024 Batches: 1 Memory Usage: 9kB
-> Seq Scan on player (cost=0.00..16.26 rows=1 width=4) (actual time=0.246..0.327 rows=1 loops=2)
Filter: ((player_name)::text = 'V Kohli'::text)
Rows Removed by Filter: 668
Planning Time: 0.218 ms
Execution Time: 55.494 ms

Creating Index:

Create index non_striker_idx on delivery_details (non_striker);

After Indexing

After creating the index on non-striker ,the execution time has been reduced to 3.633ms

-> Bitmap Heap Scan on delivery_details (cost=7.48..961.36 rows=411 width=16) (actual time=1.646..2.212 rows=5150 loops=1)
Recheck Cond: (non_striker = player.player_id)
Heap Blocks: exact=275
-> Bitmap Index Scan on delivery_details_non_striker_idx (cost=0.00..7.38 rows=411 width=0) (actual time=1.618..1.618 rows=5150 loops
Index Cond: (non_striker = player.player_id)
Planning Time: 12.808 ms
Execution Time: 3.633 ms

VII. FRONT-END(UI)

We have created a web application using PERN (PostgreSQL, Express, React, and Node. Js) stack**,** where we developed a page to insert the player details to the db and one more page to display the newly added/updated player details.

Figure 1 shows the player Details and in figure 2 we are adding a player.

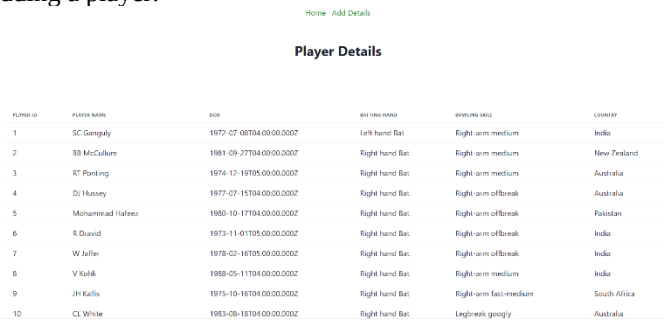


Figure 1

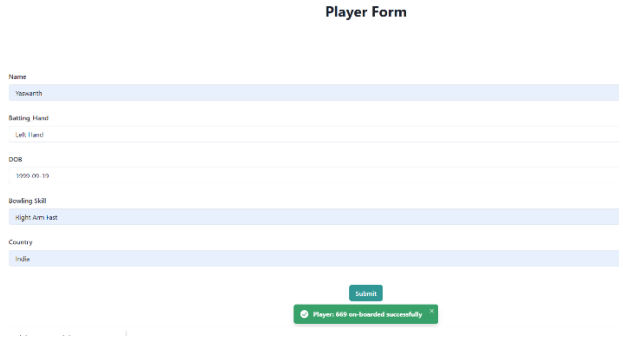


Figure 2

VIII.FUTURE SCOPE

Website: create and integrate the project into a fully functional website which would help users to access information and use it accordingly.
Visualization: with the help of visualization tools, we can analyse the data in an efficient manner which will help in making better decisions.

IX. REFERENCES

- Dataset:https://www.kaggle.com/datasets/vora1011/ipl-2008-to-2021-all-match-dataset
- Dataset 2022: https://www.kaggle.com/datasets/vora1011/ipl-2022-player-statistics