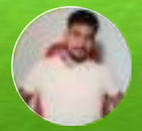
IPL Data Analysis

Dive into the insights behind the performance of IPL batsmen and bowlers.

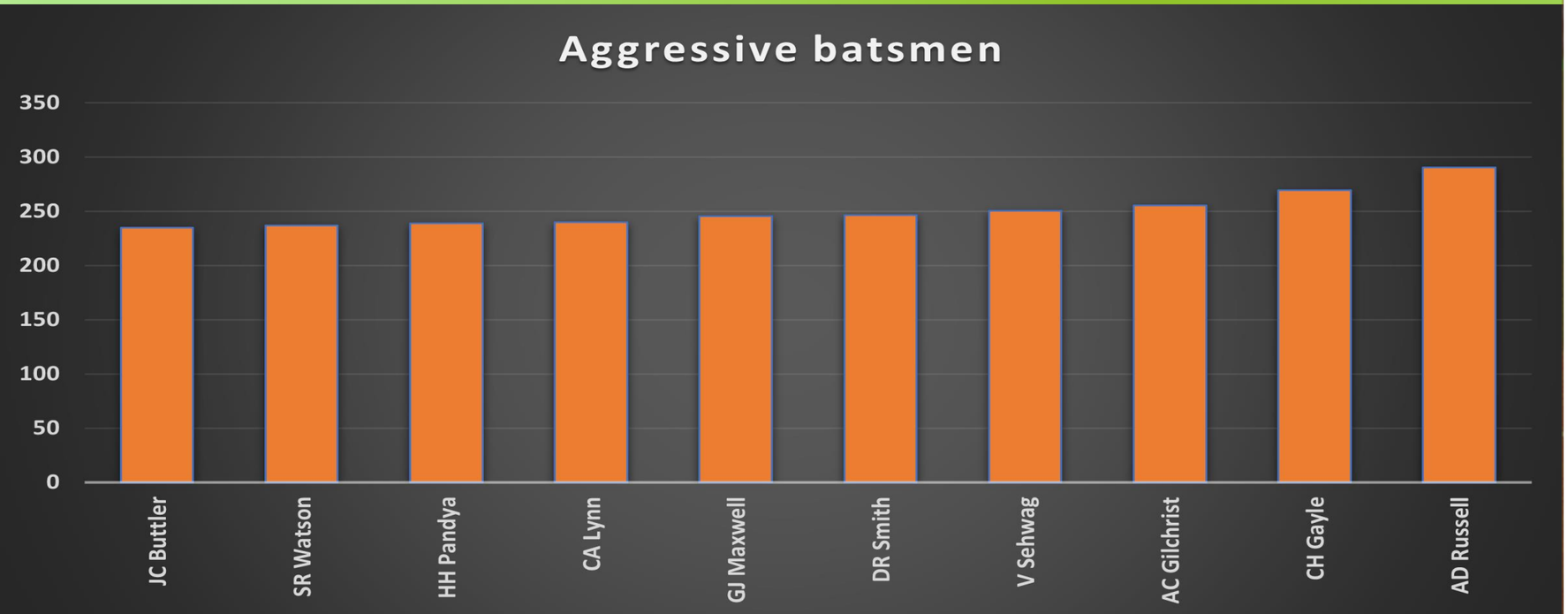


by **Asmit Kumar Pandey**

Aggressive Batsmen

```
SELECT batsman
, SUM(batsman_runs) AS total_runs, COUNT(*) - SUM(extra_runs) AS balls_faced
,  (SUM(batsman_runs)::DECIMAL / NULLIF(COUNT(*) - SUM(extra_runs), 0)) * 100 AS strike_rate
FROM deliveries
WHERE batsman_runs > 0
GROUP BY batsman
HAVING COUNT(*) - SUM(extra_runs) >= 500
ORDER BY strike_rate DESC
LIMIT 10;
```

- Essential for giving the team a strong start. Players with high strike rates and the ability to play powerplay overs effectively should be targeted.




```
39  --Aggressive Batsmen
40  SELECT batsman, SUM(batsman_runs) AS total_runs,
41         COUNT(*) - SUM(extra_runs) AS balls_faced,   --CO
42         (SUM(batsman_runs)::DECIMAL / NULLIF(COUNT(*) - S
43  FROM Deliveries
44  WHERE batsman_runs > 0
45  GROUP BY batsman
46  HAVING COUNT(*) - SUM(extra_runs) >= 500
47  ORDER BY strike_rate DESC
48  LIMIT 10;
```

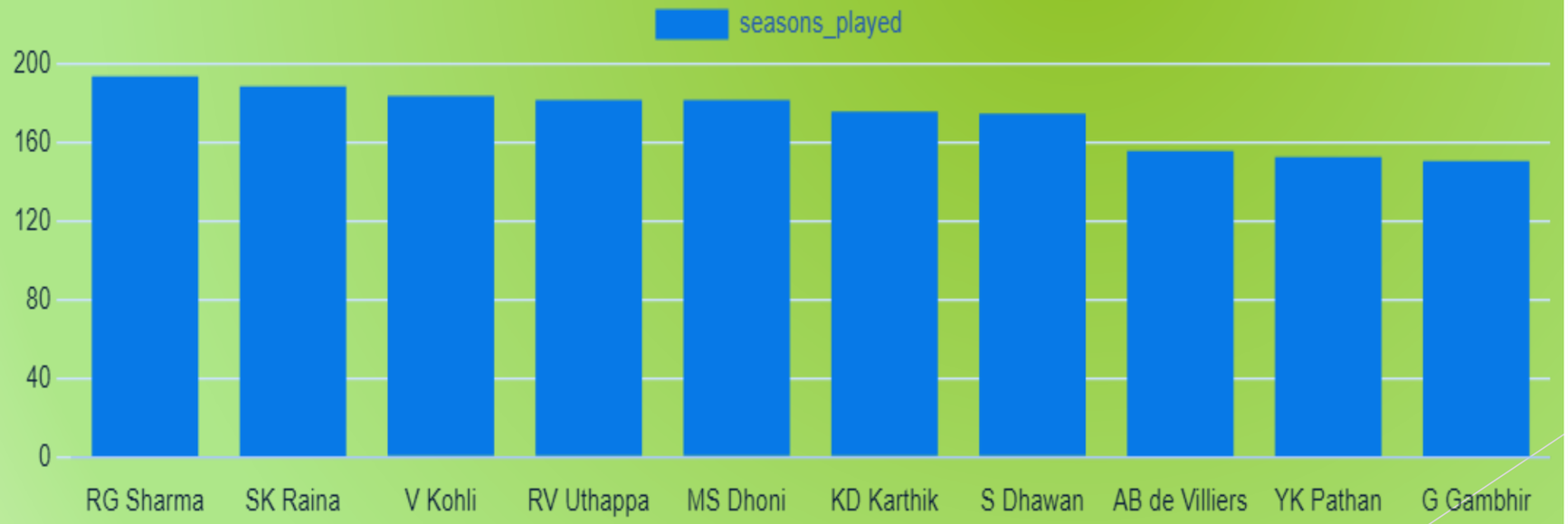
Data Output | Messages | Notifications

	batsman character varying	total_runs bigint	balls_faced bigint	strike_rate numeric
1	AD Russell	1517	522	290.6130268199233700
2	CH Gayle	4772	1771	269.4522868435911900
3	AC Gilchrist	2069	810	255.4320987654321000
4	V Sehwag	2728	1090	250.2752293577981700
5	DR Smith	2385	968	246.3842975206611600
6	GJ Maxwell	1505	613	245.5138662316476300
7	CA Lynn	1280	533	240.1500938086303900
8	HH Pandya	1349	565	238.7610619469026500

Anchor Batsmen

```
SELECT batsman
, AVG(batsman_runs) AS average_runs
, COUNT(DISTINCT id)  AS seasons_played
FROM deliveries
WHERE player_dismissed IS NOT NULL
GROUP BY batsman
HAVING COUNT(DISTINCT id) > 2
ORDER BY seasons_played DESC
LIMIT 10;
```

- These players stabilize the innings and can anchor the team during collapses. Players with consistent performance over multiple seasons are ideal.




```
50  --Anchor batsmen
51  SELECT batsman, AVG(batsman_runs) AS average_runs,
52         COUNT(DISTINCT id) AS seasons_played
53  FROM Deliveries
54  WHERE player_dismissed IS NOT NULL
55  GROUP BY batsman
56  HAVING COUNT(DISTINCT id) > 2
57  ORDER BY average_runs DESC
58  LIMIT 10;
```

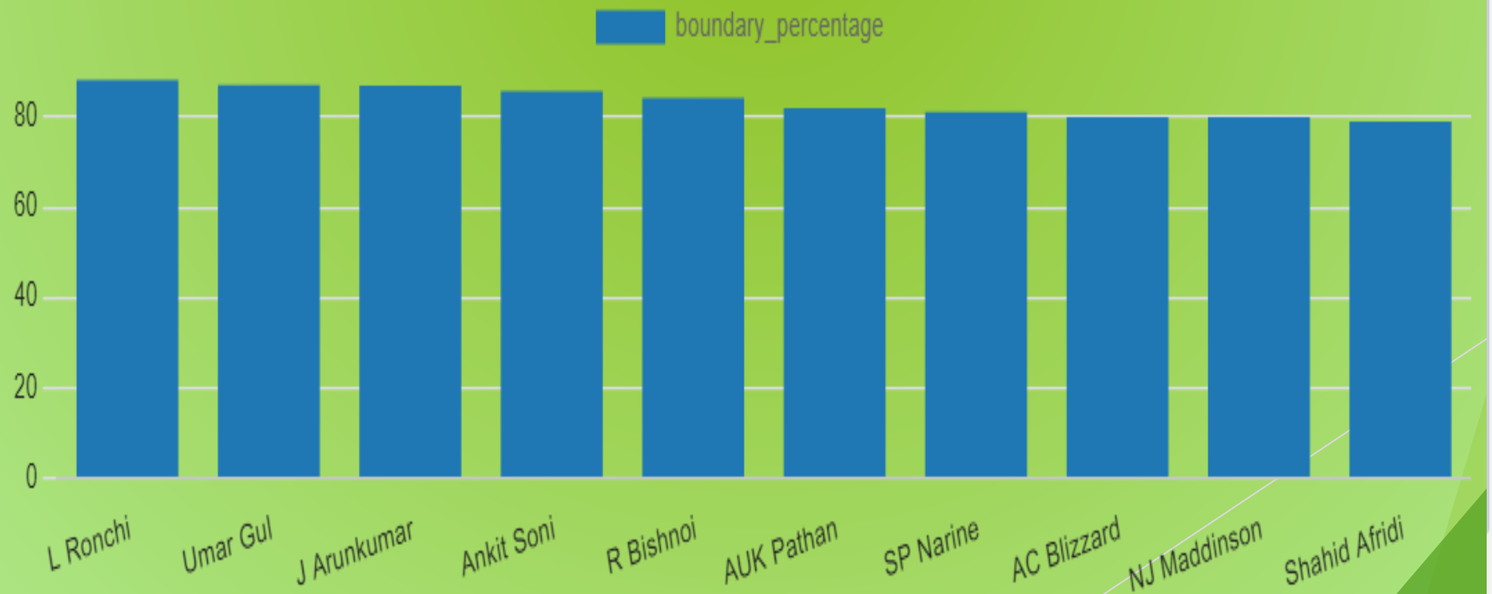
Data Output | Messages | Notifications

	batsman character varying	average_runs numeric	seasons_played bigint
1	Umar Gul	2.0526315789473684	4
2	Shahid Afridi	1.7608695652173913	9
3	AD Russell	1.7199546485260771	61
4	LJ Wright	1.6825396825396825	5
5	Abdul Samad	1.6818181818181818	7
6	KK Cooper	1.6571428571428571	12
7	K Gowtham	1.6460176991150442	19
8	Kamran Akmal	1.6410256410256410	6

Total rows: 10 of 10 | Query complete 00:00:01.591

Big Hitters

```
SELECT batsman
, SUM(batsman_runs) AS total_runs
, SUM(CASE WHEN batsman_runs = 4 THEN 1 ELSE 0 END) AS fours
, SUM(CASE WHEN batsman_runs = 6 THEN 1 ELSE 0 END) AS sixes
, (SUM(CASE WHEN batsman_runs IN (4, 6) THEN batsman_runs ELSE 0
END)::DECIMAL / SUM(batsman_runs)) * 100 AS boundary_percentage
FROM deliveries
GROUP BY batsman
HAVING COUNT(DISTINCT id) > 2
ORDER BY boundary_percentage DESC
LIMIT 10;
```



ipl_data_analysis/postgres@PostgreSQL 16* X

ipl_data_analysis/postgres@PostgreSQL 16

No limit

Query | Query History

```
1  SELECT batsman
2  , SUM(batsman_runs) AS total_runs
3  , SUM(CASE WHEN batsman_runs = 4 THEN 1 ELSE 0 END) AS fours
4  , SUM(CASE WHEN batsman_runs = 6 THEN 1 ELSE 0 END) AS sixes
5  , (SUM(CASE WHEN batsman_runs IN (4, 6) THEN batsman_runs ELSE 0
6  FROM deliveries
7  GROUP BY batsman
8  HAVING COUNT(DISTINCT id) > 2
9  ORDER BY boundary_percentage DESC
10 LIMIT 10;
```

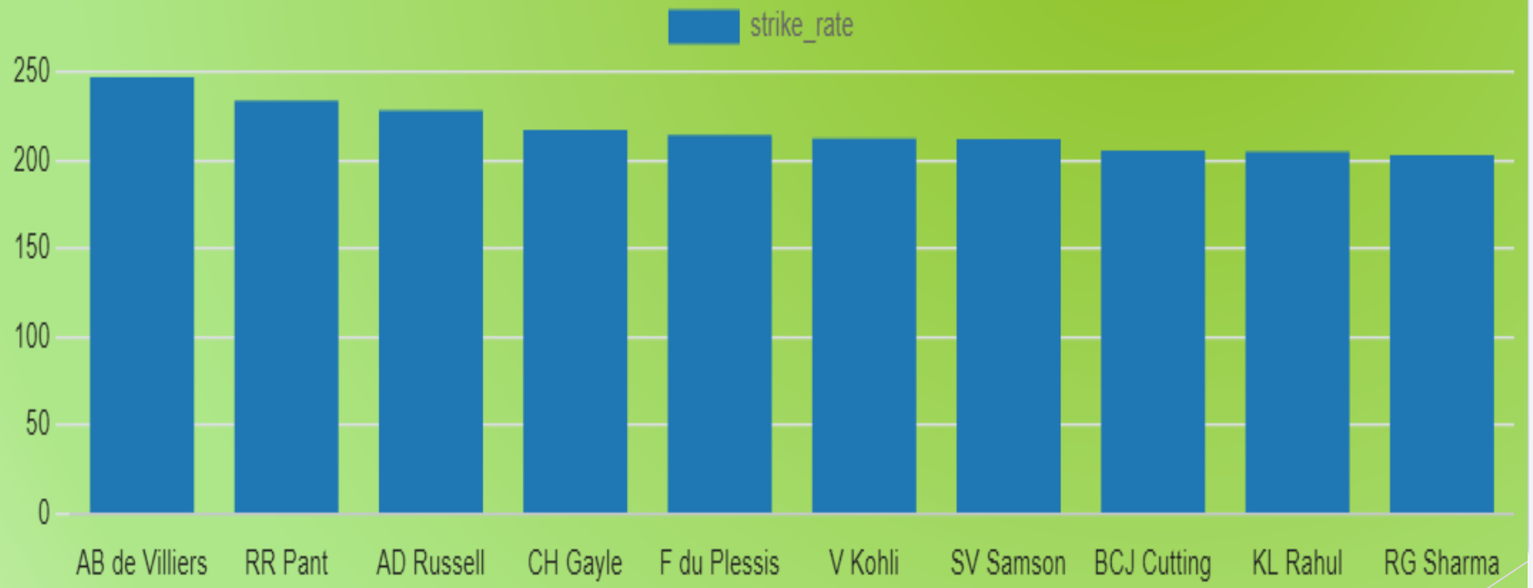
Data Output | Messages | Notifications

	batsman character varying	total_runs bigint	fours bigint	sixes bigint	boundary_percentage numeric
1	L Ronchi	34	6	1	88.23529411764705882400
2	Umar Gul	39	1	5	87.17948717948717948700
3	J Arunkumar	23	5	0	86.95652173913043478300
4	Ankit Soni	7	0	1	85.71428571428571428600
5	R Bishnoi	19	1	2	84.21052631578947368400
6	AUK Pathan	39	5	2	82.05128205128205128200
7	SP Narine	892	103	52	81.16591928251121076200
8	AC Blizzard	120	21	2	80.00000000000000000000

Total rows: 10 of 10 | Query complete 00:00:01.816

Finishers

```
SELECT batsman
,       SUM(batsman_runs) AS total_runs
,       COUNT(*) - SUM(extra_runs) AS balls_faced
,       (SUM(batsman_runs)::DECIMAL / NULLIF(COUNT(*) - SUM(extra_runs), 0)) * 100 AS
strike_rate
,       COUNT(DISTINCT id) AS matches_played
 FROM deliveries
WHERE over BETWEEN 16 AND 20
GROUP BY batsman HAVING COUNT(*) - SUM(extra_runs) >= 100
ORDER BY strike_rate DESC
LIMIT 10;
```



lpl_data_analysis/postgres@PostgreSQL 16* ×

lpl_data_analysis/postgres@PostgreSQL 16

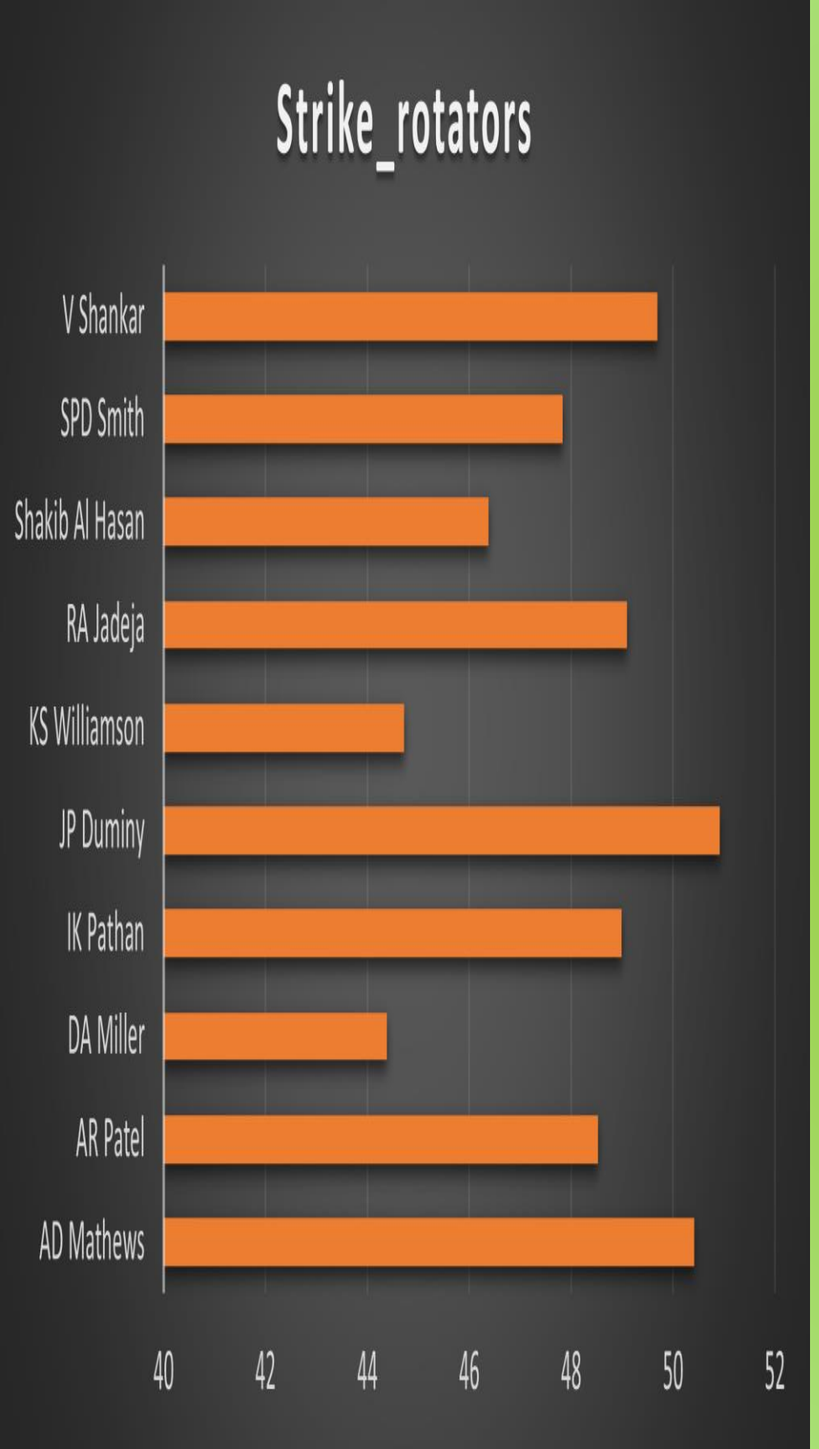
No limit

Query | Query History

```
1  --Finishers
2  SELECT batsman
3  , SUM(batsman_runs) AS total_runs
4  , COUNT(*) - SUM(extra_runs) AS balls_faced
5  , (SUM(batsman_runs)::DECIMAL / NULLIF(COUNT(*) - SUM(extra_runs), 0
6  , COUNT(DISTINCT id) AS matches_played
7  FROM deliveries
8  WHERE over BETWEEN 16 AND 20
9  GROUP BY batsman
10 HAVING COUNT(*) - SUM(extra_runs) >= 100
11 ORDER BY strike_rate DESC
12 LIMIT 10;
```

Data Output | Messages | Notifications

	batsman character varying	total_runs bigint	balls_faced bigint	strike_rate numeric	matches_played bigint
1	AB de Villiers	1254	507	247.3372781065088800	67
2	RR Pant	398	170	234.1176470588235300	28
3	AD Russell	665	291	228.5223367697594500	38
4	CH Gayle	385	177	217.5141242937853100	22
5	F du Plessis	236	110	214.5454545454545500	19
6	V Kohli	968	455	212.7472527472527500	57
7	SV Samson	310	146	212.3287671232876700	26



J P Duminy

Has a rotation rate of 50.91 making him adept at maintaining momentum.

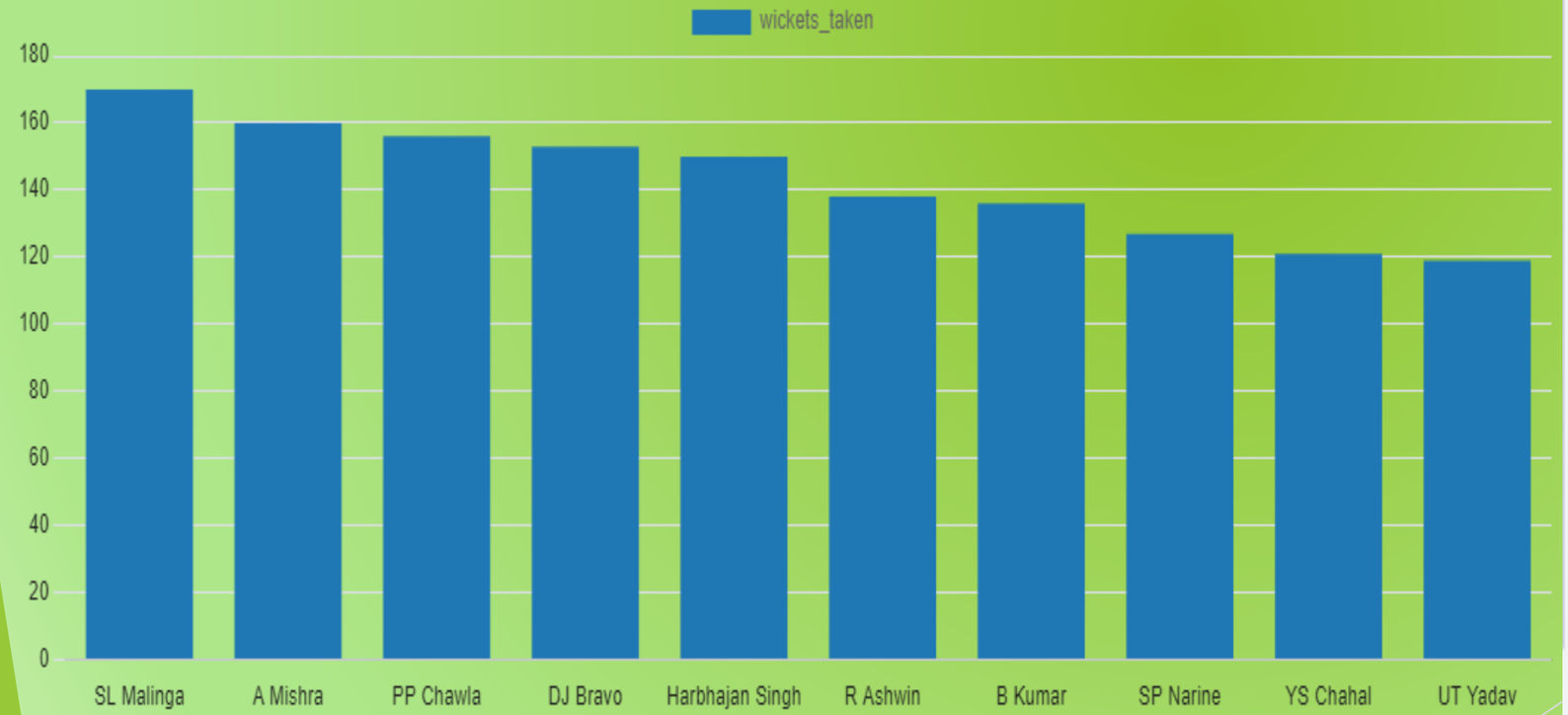
```
SELECT batsman
,       SUM(batsman_runs) AS total_runs
,       COUNT(*) - SUM(extra_runs) AS balls_faced
,       SUM(CASE WHEN batsman_runs = 1 THEN 1 ELSE 0 END) AS
singles
,       SUM(CASE WHEN batsman_runs = 2 THEN 1 ELSE 0 END) AS
doubles
,       (SUM(CASE WHEN batsman_runs IN (1, 2) THEN batsman_runs
ELSE 0 END)::DECIMAL / SUM(batsman_runs)) * 100 AS
rotation_percentage
,       (SUM(CASE WHEN batsman_runs = 1 THEN 1 ELSE 0 END) +
SUM(CASE WHEN batsman_runs = 2 THEN 1 ELSE 0 END))::DECIMAL /
NULLIF(COUNT(*) - SUM(extra_runs), 0) AS rotation_rate
FROM deliveries
GROUP BY batsman
HAVING COUNT(*) - SUM(extra_runs) >= 500
ORDER BY rotation_rate DESC
LIMIT 10;
```

Strike Rotation

Identifies batsmen who effectively rotate strike.

Wicket-Taking Bowlers

```
SELECT bowler
, COUNT(player_dismissed) AS wickets_taken
FROM deliveries
WHERE player_dismissed IS NOT NULL
AND dismissal_kind != 'run out'
AND dismissal_kind != 'NA'
GROUP BY bowler
ORDER BY wickets_taken DESC
LIMIT 10;
```



Ipl_data_analysis/postgres@PostgreSQL 16*

Ipl_data_analysis/postgres@PostgreSQL 16

No limit

Query | Query History

```
1  --Wicket-taking Bowlers
2  SELECT bowler
3  , COUNT(player_dismissed) AS wickets_taken
4  FROM deliveries
5  WHERE player_dismissed IS NOT NULL
6  AND dismissal_kind != 'run out'
7  AND dismissal_kind != 'NA'
8  GROUP BY bowler
9  ORDER BY wickets_taken DESC
10 LIMIT 10;
```

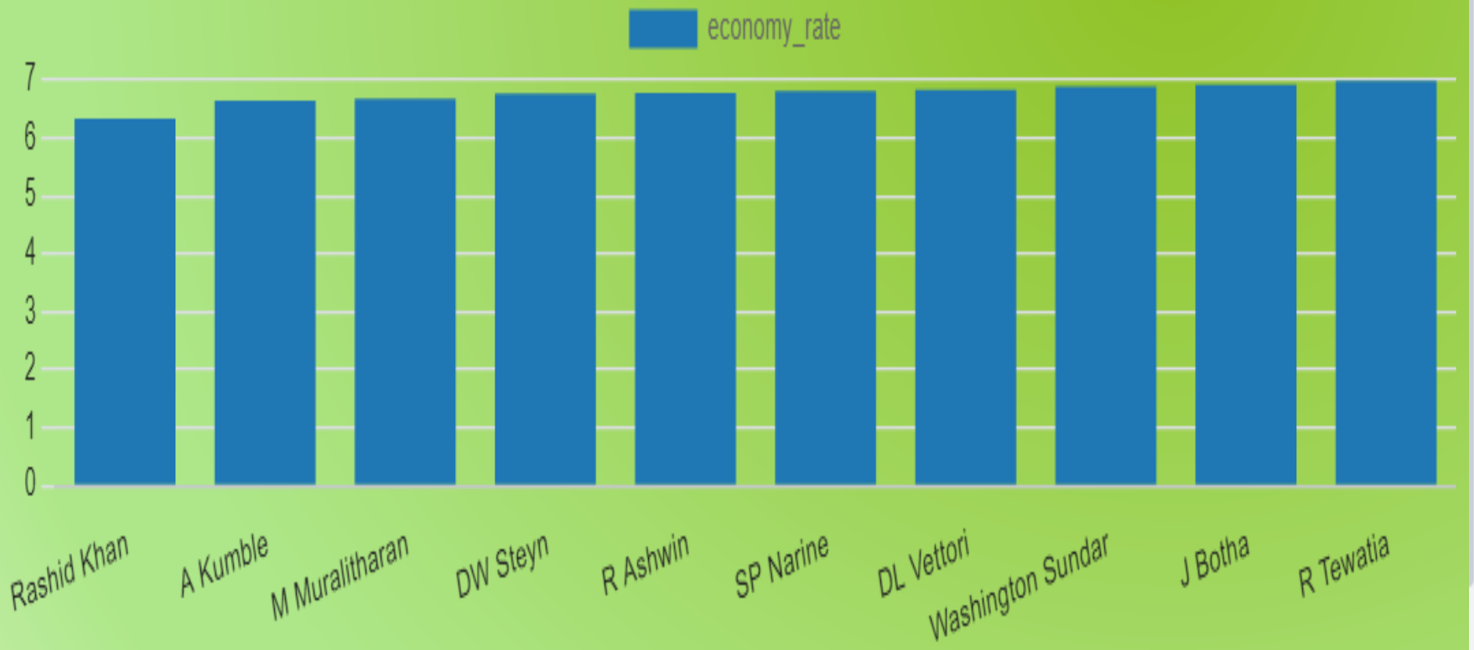
Data Output | Messages | Notifications

	bowler character varying	wickets_taken bigint
1	SL Malinga	170
2	A Mishra	160
3	PP Chawla	156
4	DJ Bravo	153
5	Harbhajan Singh	150
6	R Ashwin	138
7	B Kumar	136
8	SP Narine	127
9	YS Chahal	121

Total rows: 10 of 10 | Query complete 00:00:00.389

Economical Bowlers

```
SELECT bowler
,       SUM(total_runs) AS total_runs_conceded, COUNT(*) / 6.0 AS overs_bowled
,       (SUM(total_runs)::DECIMAL / (COUNT(*) / 6.0)) AS economy_rate
FROM deliveries
GROUP BY bowler
HAVING COUNT(*) >= 500
ORDER BY economy_rate ASC
LIMIT 10;
```



lpl_data_analysis/postgres@PostgreSQL 16*

lpl_data_analysis/postgres@PostgreSQL 16

No limit

Query | Query History

```
1  --Economical bowlers
2  SELECT bowler
3  , SUM(total_runs) AS total_runs_conceded
4  , COUNT(*) / 6.0 AS overs_bowled
5  , (SUM(total_runs)::DECIMAL / (COUNT(*) / 6.0)) AS economy_rate
6  FROM deliveries
7  GROUP BY bowler
8  HAVING COUNT(*) >= 500
9  ORDER BY economy_rate ASC
10 LIMIT 10;
```

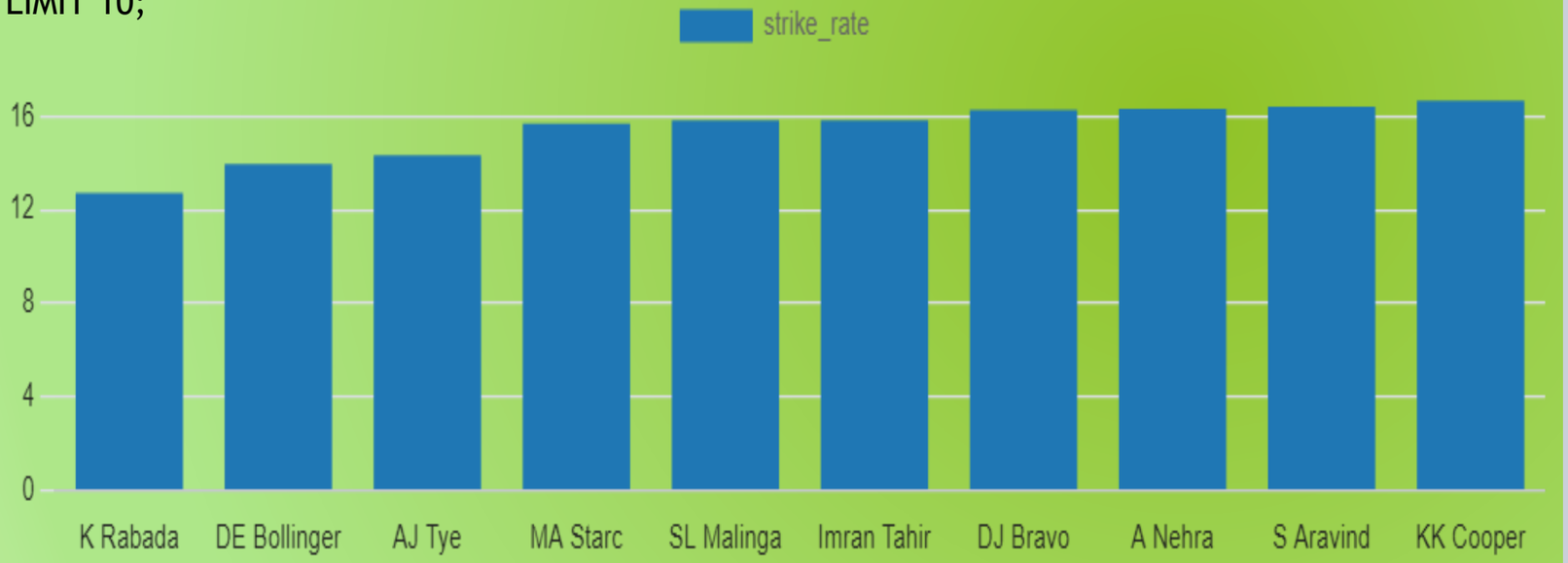
Data Output | Messages | Graph Visualiser | Notifications

	bowler character varying	total_runs_conceded bigint	overs_bowled numeric	economy_rate numeric
1	Rashid Khan	1573	248.3333333333333333	6.3342281879194631
2	A Kumble	1089	163.8333333333333333	6.6469989827060020
3	M Muralitharan	1755	262.8333333333333333	6.6772352568167406
4	DW Steyn	2568	379.3333333333333333	6.7697715289982425
5	R Ashwin	3756	554.5000000000000000	6.7736699729486023
6	SP Narine	3208	470.6666666666666667	6.8158640226628895
7	DL Vettori	894	130.8333333333333333	6.8331210191082803
8	Washington Sundar	758	110.0000000000000000	6.8909090909090909
9	J Botha	818	118.1666666666666667	6.9224259520451340

Total rows: 10 of 10 | Query complete 00:00:00.371

Best Strike Rate Bowlers

```
select bowler
,      count(ball) as balls_bowled
,      sum(is_wicket) as total_wickets
,      (count(ball)::decimal) /(nullif(sum(is_wicket),0)) as strike_rate
from deliveries
GROUP BY bowler
HAVING COUNT(ball) >=500
ORDER BY strike_rate
LIMIT 10;
```



Ipl_data_analysis/postgres@PostgreSQL 16*

Ipl_data_analysis/postgres@PostgreSQL 16

No limit

Query | Query History

```
1  --best strike rate bowlers
2  select bowler
3  , count(ball) as balls_bowled
4  , sum(is_wicket) as total_wickets
5  , (count(ball)::decimal) /(nullif(sum(is_wicket),0)) as
6  from deliveries
7  GROUP BY bowler
8  HAVING COUNT(ball) >=500
9  ORDER BY strike_rate
10 LIMIT 10;
```

Data Output | Messages | Graph Visualiser | Notifications

	bowler character varying	balls_bowled bigint	total_wickets bigint	strike_rate numeric
1	K Rabada	840	66	12.7272727272727273
2	DE Bollinger	600	43	13.9534883720930233
3	AJ Tye	645	45	14.3333333333333333
4	MA Starc	612	39	15.6923076923076923
5	SL Malinga	2974	188	15.8191489361702128
6	Imran Tahir	1314	83	15.8313253012048193
7	DJ Bravo	2846	175	16.2628571428571429
8	A Nehra	1974	121	16.3140495867768595
9	S Aravind	788	48	16.4166666666666667

Total rows: 10 of 10 | Query complete 00:00:00.545

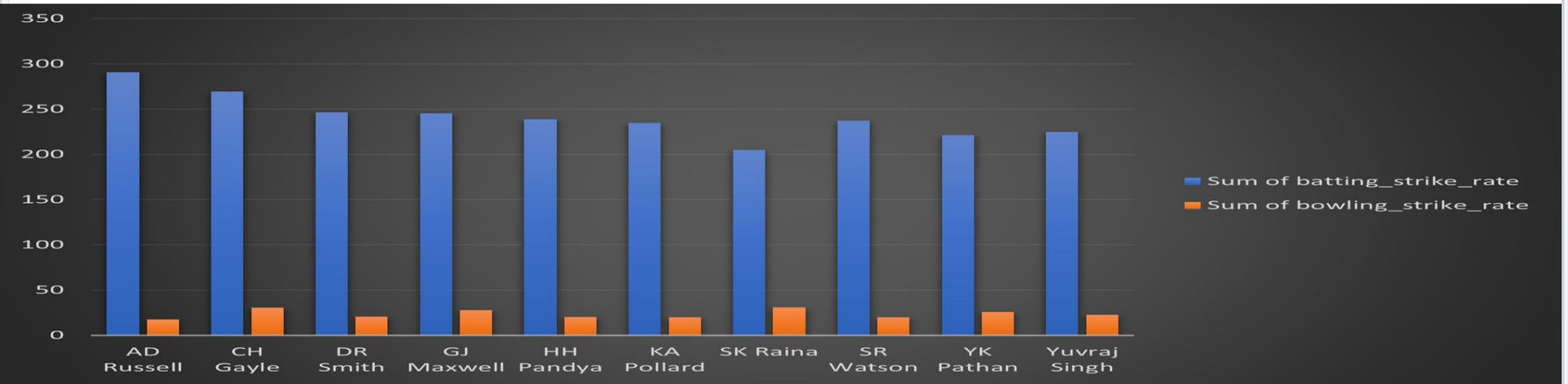
lpl_data_analysis/postgres@PostgreSQL 16*

lpl_data_analysis/postgres@PostgreSQL 16

No limit

Query | Query History

```sql
--Allrounders
SELECT b.batsman AS player
, b.balls_faced
, b.batting_strike_rate
, w.balls_bowled
, w.total_wickets
, w.bowling_strike_rate
FROM (SELECT batsman
    ,   COUNT(*) - SUM(extra_runs) AS balls_faced
    ,   (SUM(batsman_runs)::DECIMAL / NULLIF(COUNT(*) - SUM(extra_runs), 0)) * 100 AS batting_strike_rate
FROM deliveries
WHERE batsman_runs > 0
GROUP BY batsman
HAVING COUNT(*) - SUM(extra_runs) >= 500) as b
    JOIN (    SELECT bowler
    ,   COUNT(*) AS balls_bowled
    ,   SUM(is_wicket) AS total_wickets
    ,  (COUNT(*)::DECIMAL / NULLIF(SUM(is_wicket), 0)) AS bowling_strike_rate
FROM deliveries
GROUP BY bowler
HAVING COUNT(*) >= 300) as w
    ON b.batsman = w.bowler
ORDER BY (b.batting_strike_rate + w.bowling_strike_rate) / 2 DESC
LIMIT 10;
```



lpl_data_analysis/postgres@PostgreSQL 16*

lpl_data_analysis/postgres@PostgreSQL 16

No limit

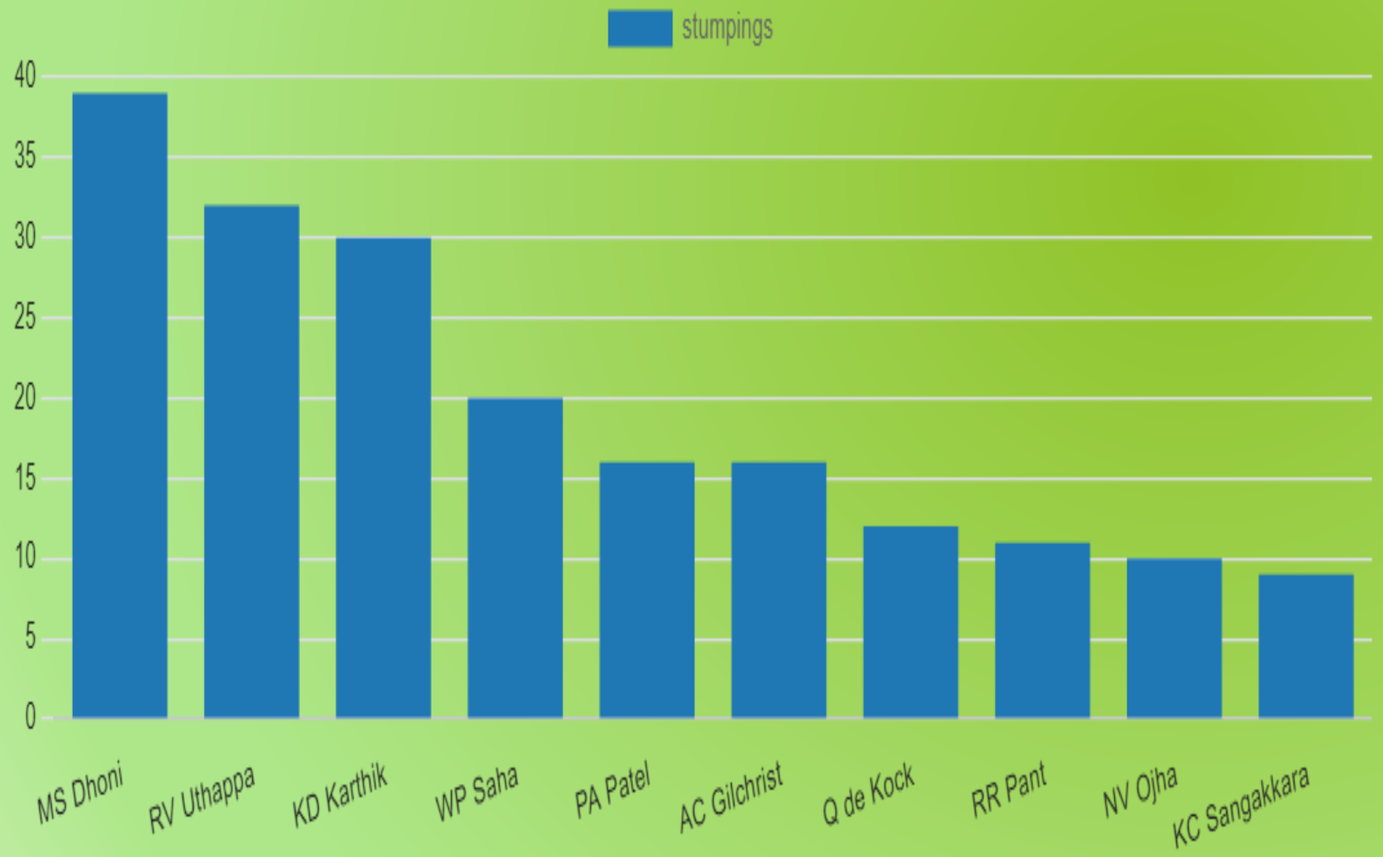
Query | Query History

Data Output | Messages | Graph Visualiser | Notifications | SQL

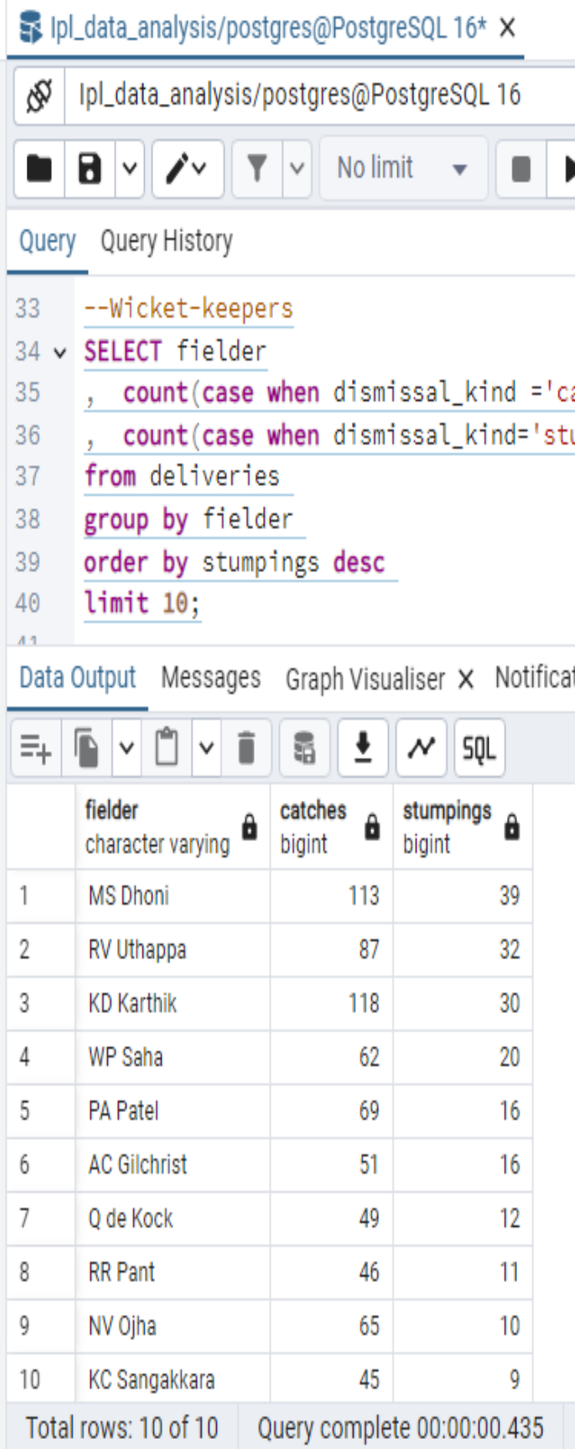
	player character varying	balls_faced bigint	batting_strike_rate numeric	balls_bowled bigint	total_wickets bigint	bowling_strike_rate numeric
1	AD Russell	522	290.6130268199233700	1186	67	17.7014925373134328
2	CH Gayle	1771	269.4522868435911900	584	19	30.7368421052631579
3	GJ Maxwell	613	245.513866231647630	558	20	27.9000000000000000
4	DR Smith	968	246.3842975206611600	557	27	20.6296296296296296
5	HH Pandya	565	238.7610619469026500	914	45	20.3111111111111111
6	SR Watson	1634	237.0869033047735600	2137	107	19.9719626168224299
7	KA Pollard	1288	234.7049689440993800	1414	71	19.9154929577464789
8	Yuvraj Singh	1224	224.6732026143790800	882	39	22.6153846153846154
9	YK Pathan	1448	221.2707182320442000	1184	46	25.7391304347826087
10	SK Raina	2619	204.9637266132111500	930	30	31.0000000000000000

Wicketkeepers

```
SELECT fielder
, count(case when dismissal_kind ='caught' then 1  else null end ) as catches
, count(case when dismissal_kind='stumped' then 1  else null end ) as stumpings
from deliveries
group by fielder
order by stumpings desc
limit 10;
```



- M S Dhoni, RV Utthapa, KD Karthik has high stumpings, indicating quick reflexes and skills.
- They also have good strike rate so that they can score quick runs.
- These players are crucial for their dual role of batting and keeping.
- These players for their skill in both wicketkeeping and potential contribution with the bat should be target.



	fielder character varying	catches bigint	stumpings bigint
1	MS Dhoni	113	39
2	RV Uthappa	87	32
3	KD Karthik	118	30
4	WP Saha	62	20
5	PA Patel	69	16
6	AC Gilchrist	51	16
7	Q de Kock	49	12
8	RR Pant	46	11
9	NV Ojha	65	10
10	KC Sangakkara	45	9

Auction Strategy

- An IPL team cannot buy more than 25 players.

- The auction plan includes 3 aggressive batsmen, 3 anchors, 3 big hitters, 3 finishers, 3 strike rotators, 3 wicket-takers, 3 economical bowlers, 3 all-rounders, and 3 wicketkeepers as total of 24 players.

```
(SELECT 'Aggressive Batsman' AS category, batsman AS player,  SUM(batsman_runs) AS total_runs,  COUNT(*) - SUM(extra_runs) AS balls_faced,       (SUM(batsman_runs)::DECIMAL / NULLIF(COUNT(*) - SUM(extra_runs), 0)) * 100 AS strike_rate
FROM deliveries
WHERE batsman_runs > 0
GROUP BY batsman
HAVING COUNT(*) - SUM(extra_runs) >= 500
ORDER BY strike_rate DESC
LIMIT 3)
UNION ALL
(SELECT 'Anchor Batsman' AS category, batsman AS player,  AVG(batsman_runs) AS average_runs,  COUNT(DISTINCT id) AS seasons_played,   NULL AS strike_rate
FROM deliveries
WHERE player_dismissed IS NOT NULL
GROUP BY batsman
HAVING COUNT(DISTINCT id) > 2
ORDER BY seasons_played DESCLIMIT 3)
UNION ALL
(SELECT 'Big Hitter' AS category, batsman AS player,  SUM(batsman_runs) AS total_runs,  SUM(CASE WHEN batsman_runs = 4 THEN 1 ELSE 0 END) AS fours,   SUM(CASE WHEN batsman_runs = 6 THEN 1 ELSE 0 END) AS sixes
FROM deliveries
GROUP BY batsman
HAVING COUNT(DISTINCT id) > 2
ORDER BY sixes DESC
LIMIT 3)
UNION ALL
(SELECT 'Finisher' AS category, batsman AS player,  SUM(batsman_runs) AS total_runs,   COUNT(*) - SUM(extra_runs) AS balls_faced,  (SUM(batsman_runs)::DECIMAL / NULLIF(COUNT(*) - SUM(extra_runs), 0)) * 100 AS strike_rate
FROM deliveries
WHERE over BETWEEN 16 AND 20
GROUP BY batsman
HAVING COUNT(*) - SUM(extra_runs) >= 100
ORDER BY strike_rate DESC
LIMIT 3)
UNION ALL
(SELECT 'Strike Rotator' AS category, batsman AS player,   SUM(batsman_runs) AS total_runs,   SUM(CASE WHEN batsman_runs = 1 THEN 1 ELSE 0 END) AS singles,  SUM(CASE WHEN batsman_runs = 2 THEN 1 ELSE 0 END) AS doubles
 FROM deliveries
GROUP BY batsman
HAVING COUNT(*) - SUM(extra_runs) >= 500
ORDER BY singles DESC
LIMIT 3)UNION ALL
(SELECT 'Wicket-Taking Bowler' AS category, bowler AS player,  COUNT(player_dismissed) AS wickets_taken,  NULL AS balls_faced,  NULL AS strike_rate
FROM deliveries
WHERE player_dismissed IS NOT NULL
GROUP BY bowler
ORDER BY wickets_taken DESC
LIMIT 3)
UNION ALL
(SELECT 'Economical Bowler' AS category, bowler AS player,   SUM(total_runs) AS total_runs_conceded,   COUNT(*) / 6.0 AS overs_bowled,   (SUM(total_runs)::DECIMAL / (COUNT(*) / 6.0)) AS economy_rate
FROM deliveries
GROUP BY bowler
HAVING COUNT(*) >= 500
ORDER BY economy_rate ASC
LIMIT 3)
UNION ALL
(SELECT 'All-Rounder' AS category, b.batsman AS player,   b.batting_strike_rate AS strike_rate,    w.total_wickets,    w.bowling_strike_rateFROM    (SELECT batsman,   COUNT(*) - SUM(extra_runs) AS balls_faced,   (SUM(batsman_runs)::DECIMAL / NULLIF(COUNT(*) - SUM(extra_runs), 0)) * 100 AS
batting_strike_rate
 FROM deliveries
WHERE batsman_runs > 0
 GROUP BY batsman
HAVING COUNT(*) - SUM(extra_runs) >= 500
ORDER BY batting_strike_rate DESC    LIMIT 5) as bJOIN     (SELECT bowler,          COUNT(*) AS balls_bowled,          SUM(is_wicket) AS total_wickets,          (COUNT(*)::DECIMAL / NULLIF(SUM(is_wicket), 0)) AS bowling_strike_rate    FROM ipl_ball    GROUP BY bowler    HAVING COUNT(*) >= 300    ORDER
BY bowling_strike_rate ASC    LIMIT 5) as w ON b.batsman = w.bowlerORDER BY (b.batting_strike_rate + w.bowling_strike_rate) / 2 DESCLIMIT 3)UNION ALL-- Wicketkeepers(SELECT 'Wicketkeeper' AS category, fielder AS player,       count(case when dismissal_kind ='caught' then 1 else null end ) AS
catches,       count(case when dismissal_kind='stumped' then 1 else null end ) AS stumpings,       NULL AS strike_rate FROM deliveries GROUP BY fielder ORDER BY stumpings DESC LIMIT 3);
```

Additional

- **1.There are 33 cities hosting IPL matches.**

```
SELECT COUNT(DISTINCT city) AS cities_hosted_ipl FROM ipl_matches;
```

- **2.Created table’deliveries’ using table ipl_ball**

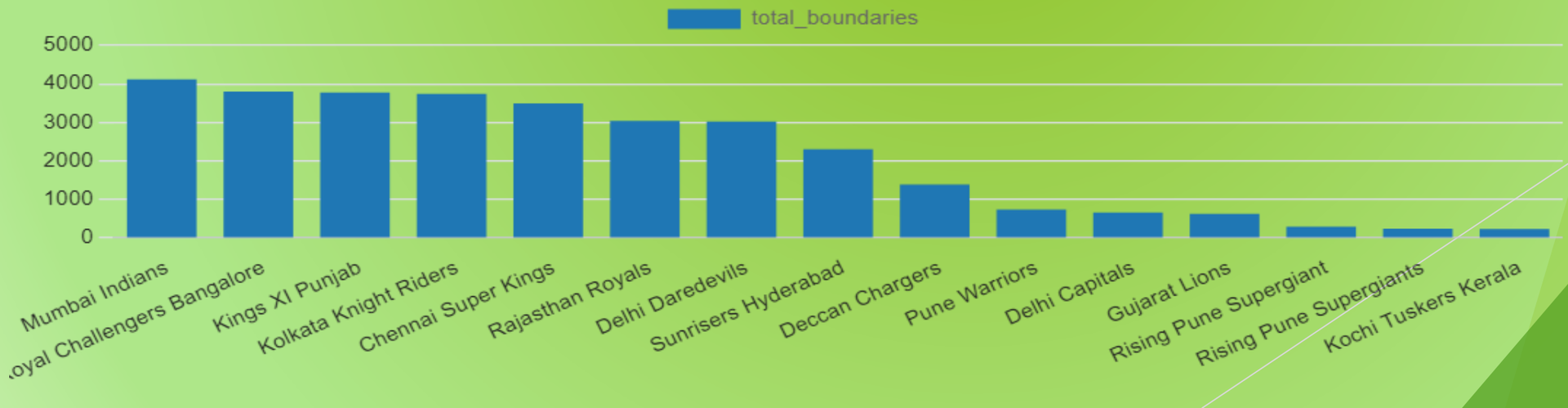
```
CREATE TABLE deliveries AS SELECT id, inning, over, ball, batsman, non_striker, bowler, batsman_runs, extra_runs, total_runs, is_wicket, dismissal_kind, player_dismissed, fielder, extras_type, batting_team, bowling_team,
CASE
WHEN total_runs >= 4 THEN 'boundary'
WHEN total_runs = 0 THEN 'dot' ELSE 'other' END AS ball_result
FROM deliveries;
```

- **3. Total boundries hit are 31468 and total dot balls are 67841.**

```
SELECT
SUM(CASE WHEN ball_result = 'boundary' THEN 1 ELSE 0 END) AS total_boundaries,
SUM(CASE WHEN ball_result = 'dot' THEN 1 ELSE 0 END) AS total_dot_balls
FROM deliveries;
```

- **4. Total boundaries hit by each team.**

```
SELECT batting_team,
SUM(CASE WHEN ball_result = 'boundary' THEN 1 ELSE 0 END) AS total_boundaries
FROM deliveries
GROUP BY batting_team
ORDER BY total_boundaries DESC;
```



Additional

- **5. Most dot balls bowled by team Mumbai Indians.**

```
SELECT  bowling_team,
SUM(CASE WHEN ball_result = 'dot' THEN 1 ELSE 0 END) AS total_dot_balls
FROM deliveries
GROUP BY bowling_team
ORDER BY total_dot_balls DESC;
```

- **6. Most number of dismissal kind are caught.**

```
SELECT     dismissal_kind,     COUNT(*) AS total_dismissals
FROM deliveries
WHERE dismissal_kind IS NOT NULL
AND dismissal_kind <> 'NA'
GROUP BY dismissal_kind
ORDER BY total_dismissals DESC;
```

- **7. Most extra runs are given by SL malinga.**

```
SELECT     bowler,
SUM(extra_runs) AS total_extra_runs
FROM deliveries
GROUP BY bowler
ORDER BY total_extra_runs DESC
LIMIT 5;
```

- **8. Created table deliveries3 using table deliveries2 and ipl_matches.**

```
CREATE TABLE deliveries3 AS  SELECT  d.*,     m.venue,    m.date AS match_date
FROM     deliveries as d
JOIN     ipl_matches as m ON     d.id = m.id;
```

- **9. Most runs scored on venue Eden Gardens followed by Wankhede Stadium.**

```
SELECT venue, SUM(total_runs) AS total_runs_scored
FROM deliveries3
GROUP BY venue
ORDER BY total_runs_scored DESC;
```

10. Most runs(2885) scored in ipl season 2018 folloed by runs(2651) scored in ipl season 2019 at Eden Gardens.

```
SELECT EXTRACT(YEAR FROM match_date) AS year,       SUM(total_runs) AS total_runs_scored
FROM deliveries3
WHERE venue = 'Eden Gardens'
GROUP BY year
ORDER BY total_runs_scored DESC;
```

Conclusion

- After a deep analysis I can say that To build a competitive and balanced IPL team, it is essential to focus on a combination of aggressive and anchor batsmen, effective finishers, big-hitters, and consistent rotators of strike. The team should also have a mix of bowlers who can perform in different match situations and all-rounders who provide flexibility.
- I tried a well-thought-out auction strategy to involve detailed data analysis of players' based on performances and potential impact. By strategically allocating the budget and making informed bidding decisions, the new IPL franchise can assemble a squad capable of competing at the highest level.

Any suggestion is welcome.

THANKS☺