Creating ipl_ball tables and copy data from CSV file

```
-- CREATING THE TABLE OF IPL_BALL
create table ipl_ball (
    id bigint,
    inning int,
    over int,
    ball int,
    batsman varchar(255),
    non_striker varchar(255),
    bowler varchar(255),
    batsman_runs float,
    extra_runs int,
    total_runs float,
    is_wicket int,
    dismissal_kind varchar(255),
    player_dismissed varchar(255),
    fielder varchar(255),
    extras_type varchar(255),
    batting_team varchar(255),
    bowling_team varchar(255)
);


-- COPYING DATA FROM CSV FILE
COPY ipl_ball (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)
FROM 'C:/Program Files/PostgreSQL/16/datacsv/IPL Dataset/IPL_Ball.csv' DELIMITER ',' CSV HEADER;

-- Retrieving all data from the table
select * from ipl_ball
```

Creating ipl_matches tables and copy data from CSV file

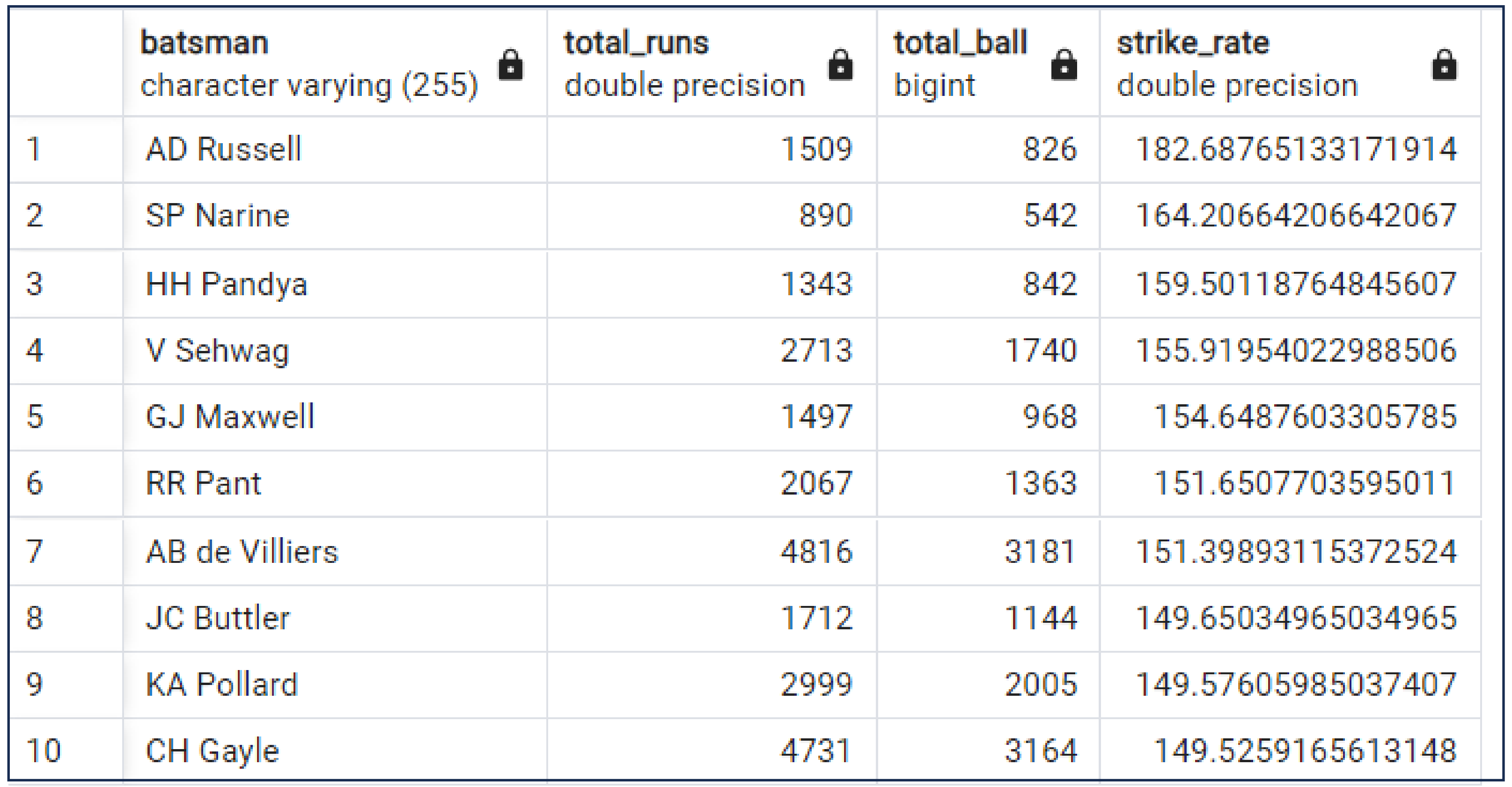
```
-- CREATING THE TABLE OF IPL_MATCHES
create table ipl_matches (
    id bigint,
    city varchar(255),
    match_date date,
    player_of_match varchar(255),
    venue varchar(255),
    neutral_venue int,
    team1 varchar(255),
    team2 varchar(255),
    toss_winner varchar(255),
    toss_decision varchar(255),
    winner varchar(255),
    result varchar(255),
    result_margin int,
    eliminator varchar(255),
    method varchar(255),
    umpire1 varchar(255),
    umpire2 varchar(255)
);

-- COPYING DATA FROM CSV FILE
COPY ipl_matches (id, city, match_date, player_of_match, venue, neutral_venue, team1, team2, toss_winner, toss_decision, winner, result,
result_margin, eliminator, method, umpire1, umpire2)
FROM 'C:/Program Files/PostgreSQL/16/datacsv/IPL Dataset/IPL_matches.csv' DELIMITER ',' CSV HEADER;


-- Retrieving all data from the table
select * from ipl_matches
```

Joining the two above tables for convenience

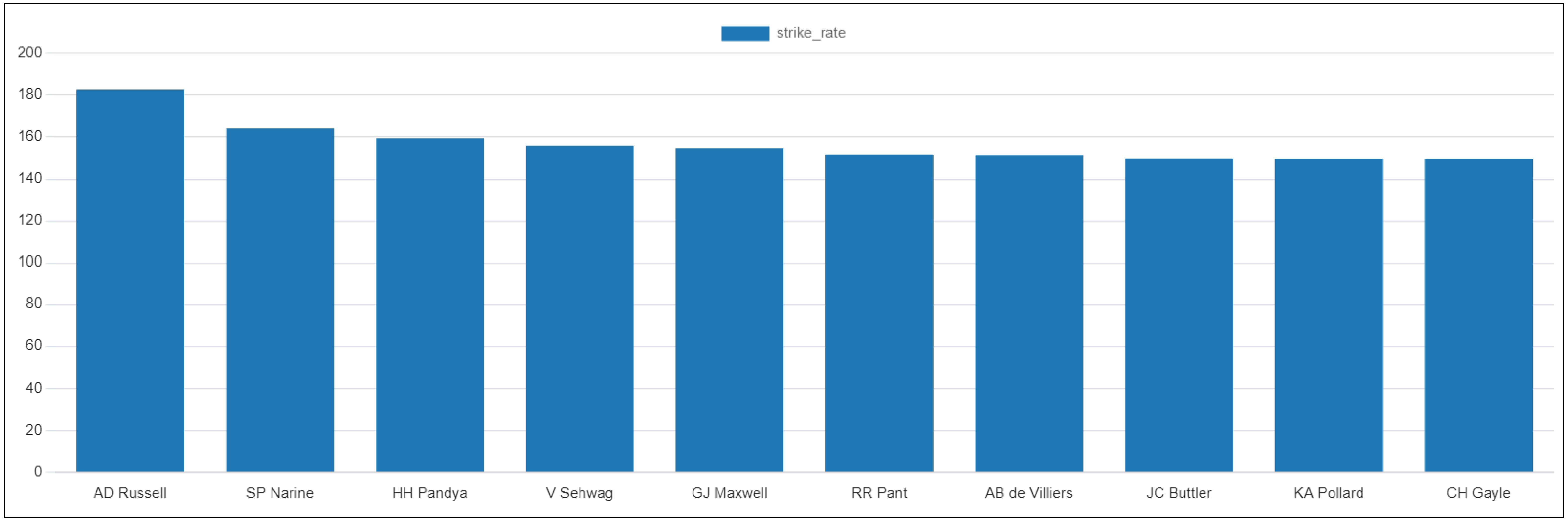
```
-- JOINING THE TWO TABLES FOR CONVENIENCE
create table all_table as
select
a.id, a.inning,  a.over,     a.ball,      a.batsman,      a.non_striker,  a.bowler, a.batsman_runs, a.extra_runs, a.total_runs,
a.is_wicket,
a.dismissal_kind,    a.player_dismissed,a.fielder,  a.extras_type, a.batting_team,      a.bowling_team,
b.city,      b.match_date, b.player_of_match, b.venue,  b.neutral_venue,    b.team1,  b.team2,  b.toss_winner,
        b.toss_decision,
b.winner, b.result,   b.result_margin,     b.eliminator,    b.method,       b.umpire1,      b.umpire2
from ipl_ball as a join ipl_matches as b on a.id=b.id


select * from all_table
```

TASK 1
(Finding the list of batsman who has high strike rate and also faced at least 500 balls)

```
select *, (table1.total_runs/table1.total_ball)*100 as "strike_rate"
from(select batsman, sum(batsman_runs) as "total_runs",
                        count(ball) as "total_ball"
    from ipl_ball where extras_type not in ('wides', 'noballs')
group by batsman) as table1
where total_ball>=500 order by strike_rate desc limit 10
```



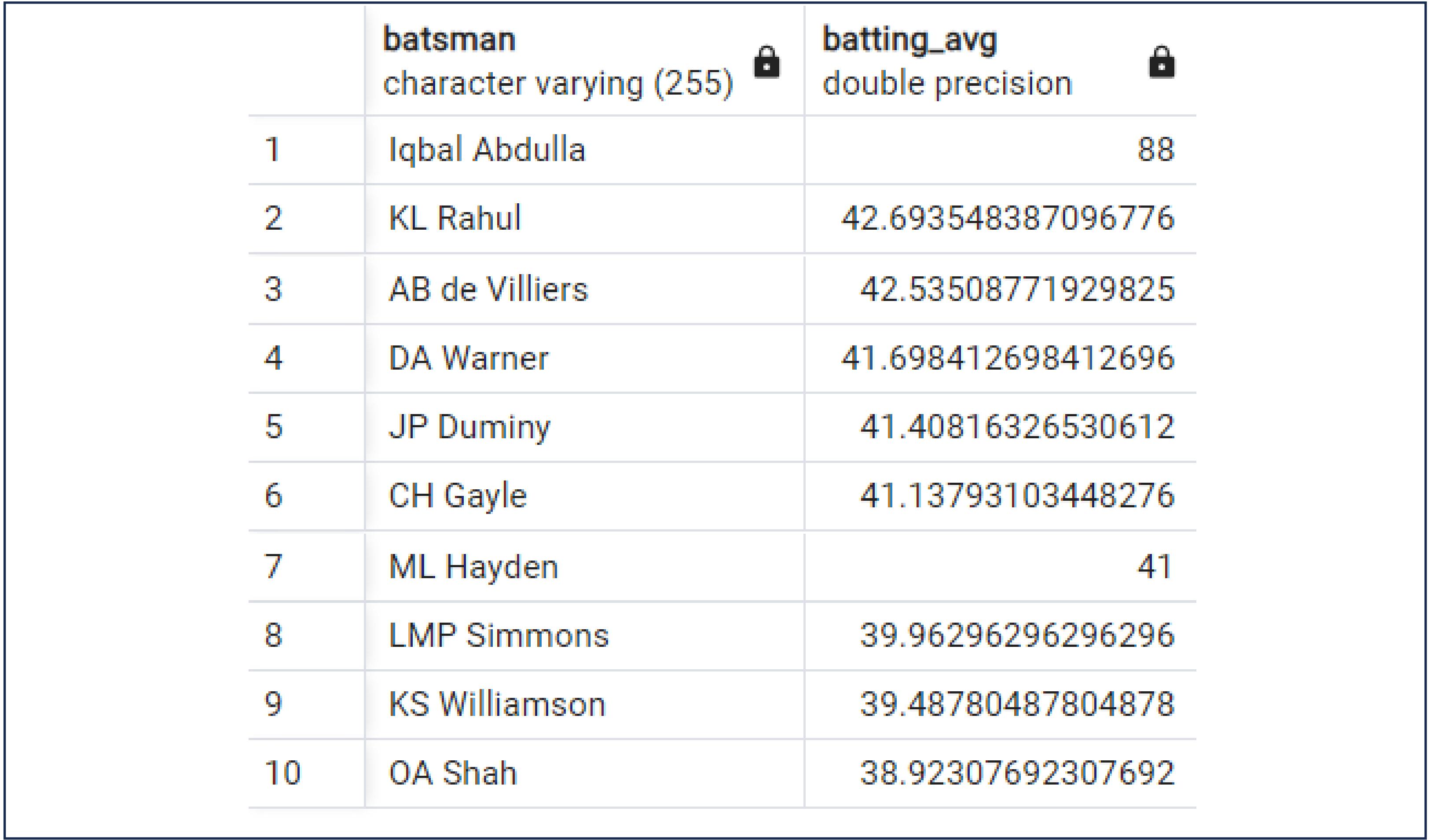
	batsman character varying (255)	total_runs double precision	total_ball bigint	strike_rate double precision
1	AD Russell	1509	826	182.68765133171914
2	SP Narine	890	542	164.20664206642067
3	HH Pandya	1343	842	159.50118764845607
4	V Sehwag	2713	1740	155.91954022988506
5	GJ Maxwell	1497	968	154.6487603305785
6	RR Pant	2067	1363	151.6507703595011
7	AB de Villiers	4816	3181	151.39893115372524
8	JC Buttler	1712	1144	149.65034965034965
9	KA Pollard	2999	2005	149.57605985037407
10	CH Gayle	4731	3164	149.5259165613148



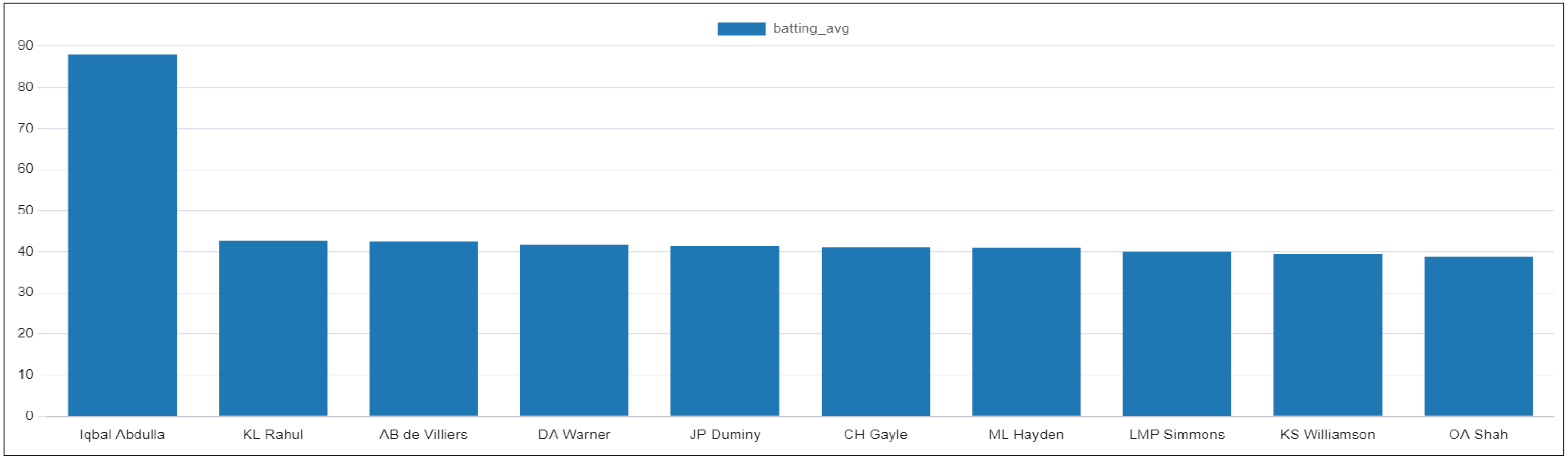
TASK 2

(Finding the list of anchor batsman or the batsman having good average Who have played more than 2 season)

```
select
      batsman,
      sum(batsman_runs)/sum(case when is_wicket=1 then 1 else 0 end) as
batting_avg
from all_table
group by batsman
      having count(distinct extract(year from match_date))>2
order by batting_avg desc limit 10
```

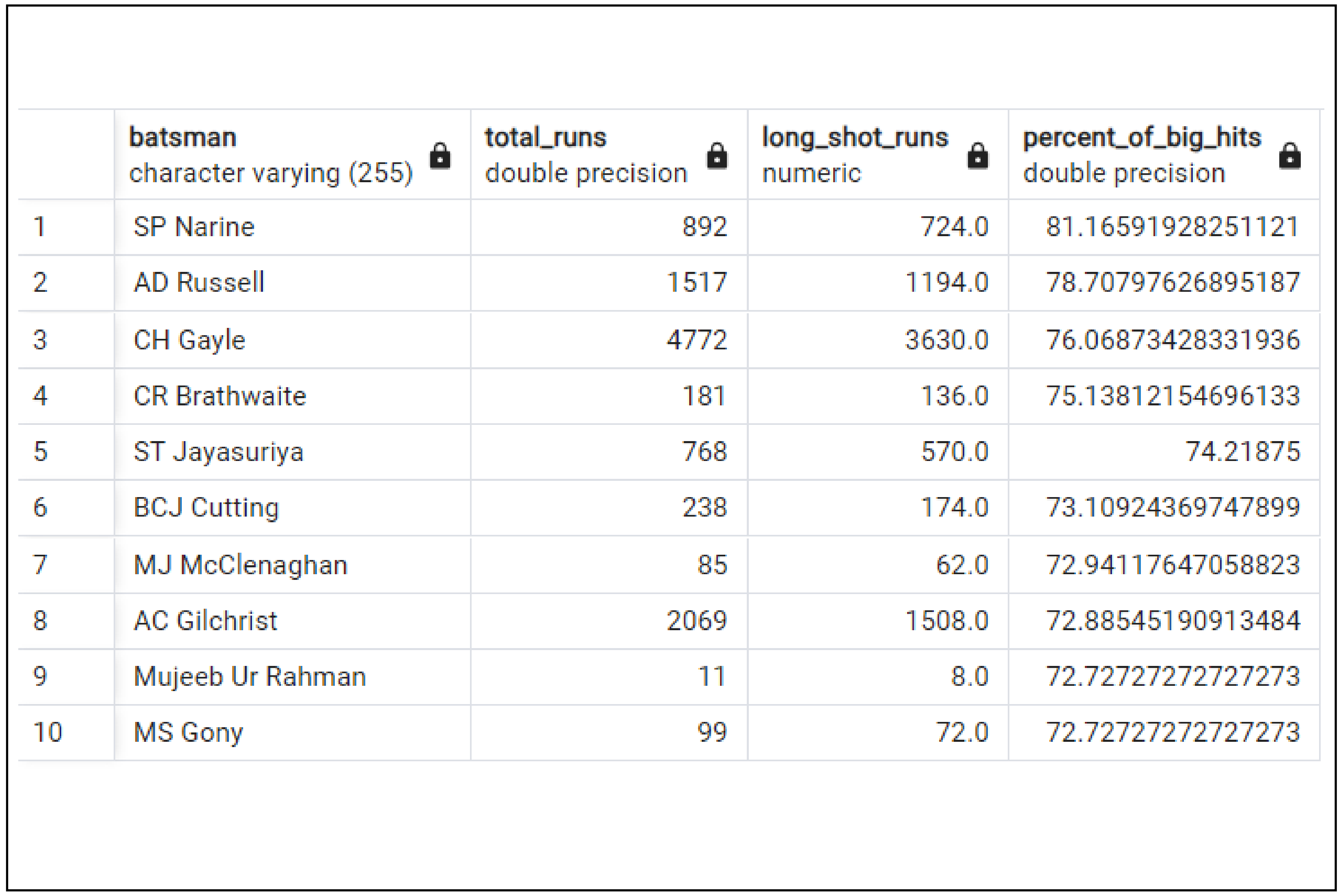


	batsman character varying (255)	batting_avg double precision
1	Iqbal Abdulla	88
2	KL Rahul	42.693548387096776
3	AB de Villiers	42.53508771929825
4	DA Warner	41.698412698412696
5	JP Duminy	41.40816326530612
6	CH Gayle	41.13793103448276
7	ML Hayden	41
8	LMP Simmons	39.96296296296296
9	KS Williamson	39.48780487804878
10	OA Shah	38.92307692307692

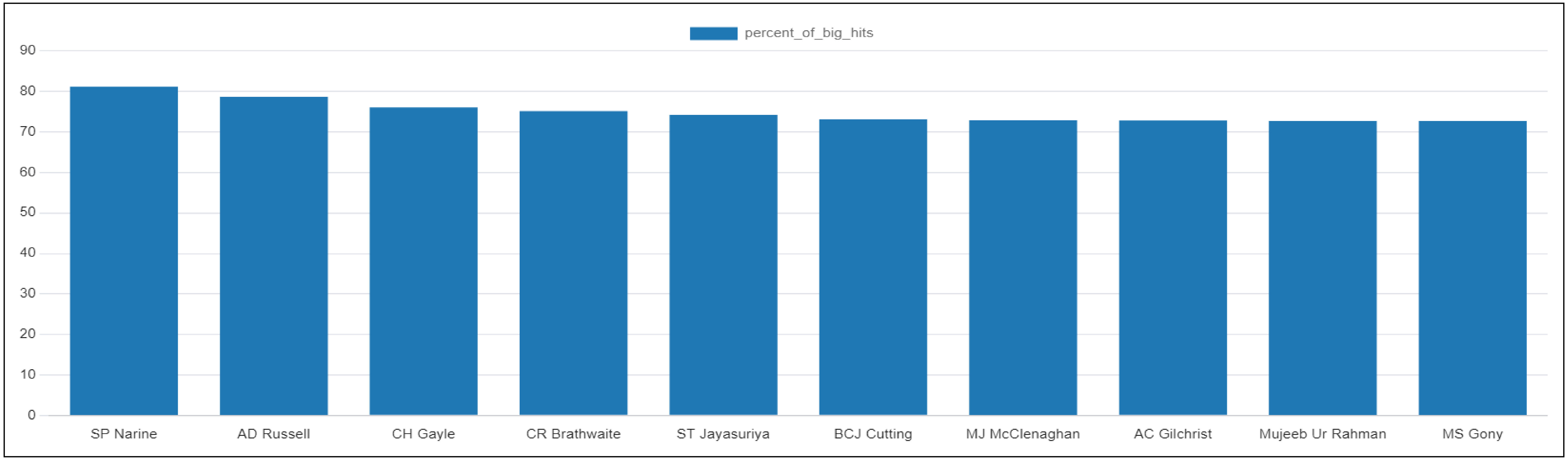


TASK 3
(Finding the list of big hitters)

```
select batsman, sum(batsman_runs) as total_runs,
                    (sum(case when batsman_runs=4 then 1 else 0 end)*4.0
                    +sum(case when batsman_runs=6 then 1 else 0 end)*6.0) as
long_shot_runs,
                    (sum(case when batsman_runs=4 then 1 else 0 end)*4.0
                    +sum(case when batsman_runs=6 then 1 else 0
end)*6.0)/sum(batsman_runs)*100.0 as percent_of_big_hits
from all_table
group by batsman
having (sum(batsman_runs) >0)
AND
count(distinct extract(year from match_date))>2
order by percent_of_big_hits desc limit 10
```

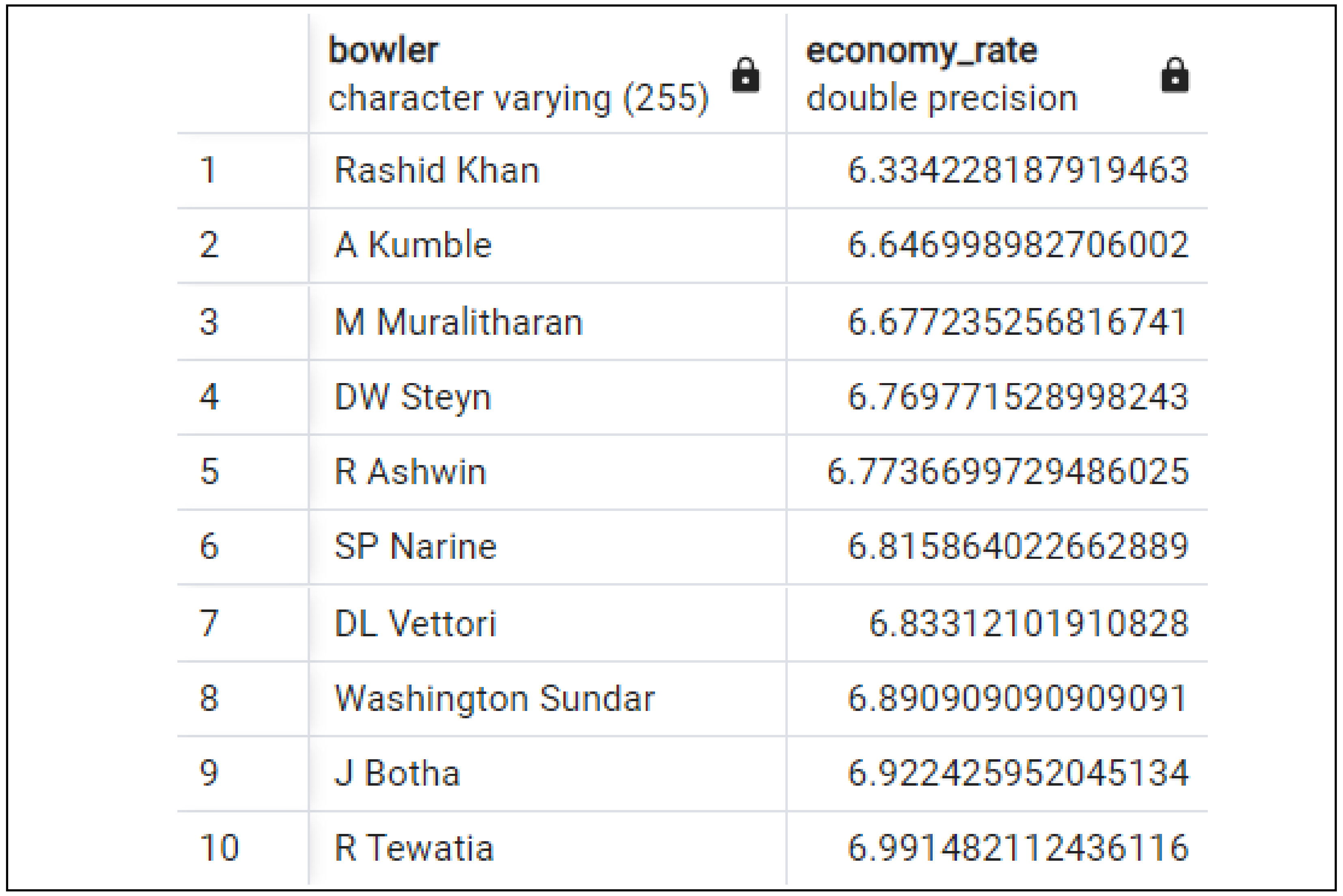


	batsman character varying (255)	total_runs double precision	long_shot_runs numeric	percent_of_big_hits double precision
1	SP Narine	892	724.0	81.16591928251121
2	AD Russell	1517	1194.0	78.70797626895187
3	CH Gayle	4772	3630.0	76.06873428331936
4	CR Brathwaite	181	136.0	75.13812154696133
5	ST Jayasuriya	768	570.0	74.21875
6	BCJ Cutting	238	174.0	73.10924369747899
7	MJ McClenaghan	85	62.0	72.94117647058823
8	AC Gilchrist	2069	1508.0	72.88545190913484
9	Mujeeb Ur Rahman	11	8.0	72.72727272727273
10	MS Gony	99	72.0	72.72727272727273

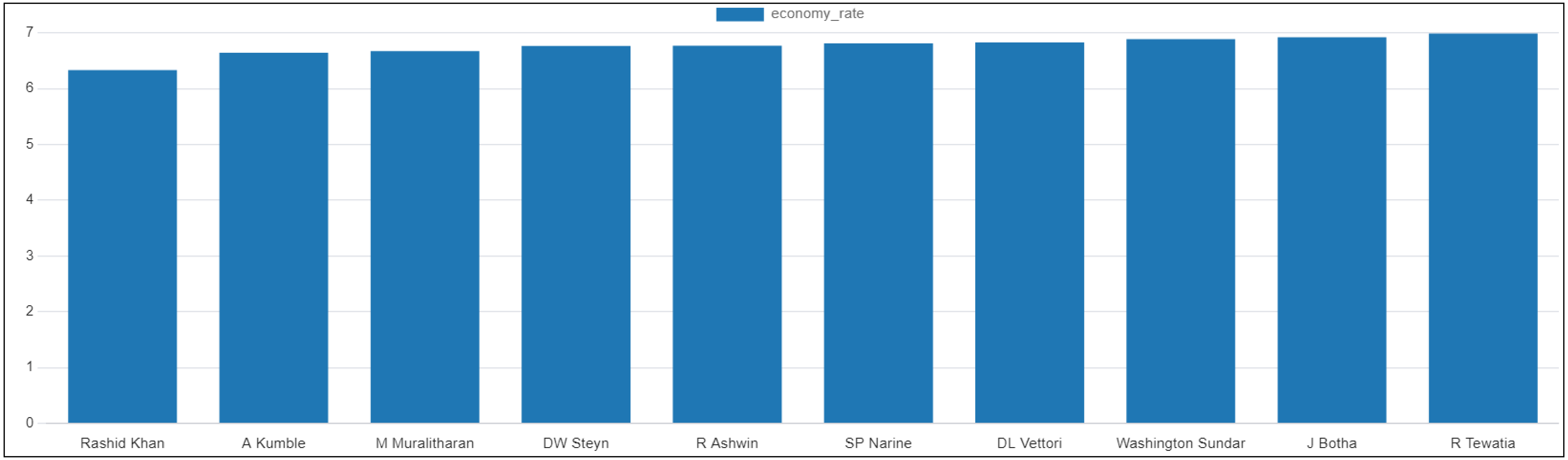


TASK 4
(Finding the bowler having good economy rate)

```
select bowler, (total_runs/(total_deliveries/6.0)) as economy_rate
from (select bowler, sum(total_runs) as total_runs,
                                    count(ball) as total_deliveries
from all_table group by bowler)
where total_deliveries>=500
order by economy_rate limit 10
```

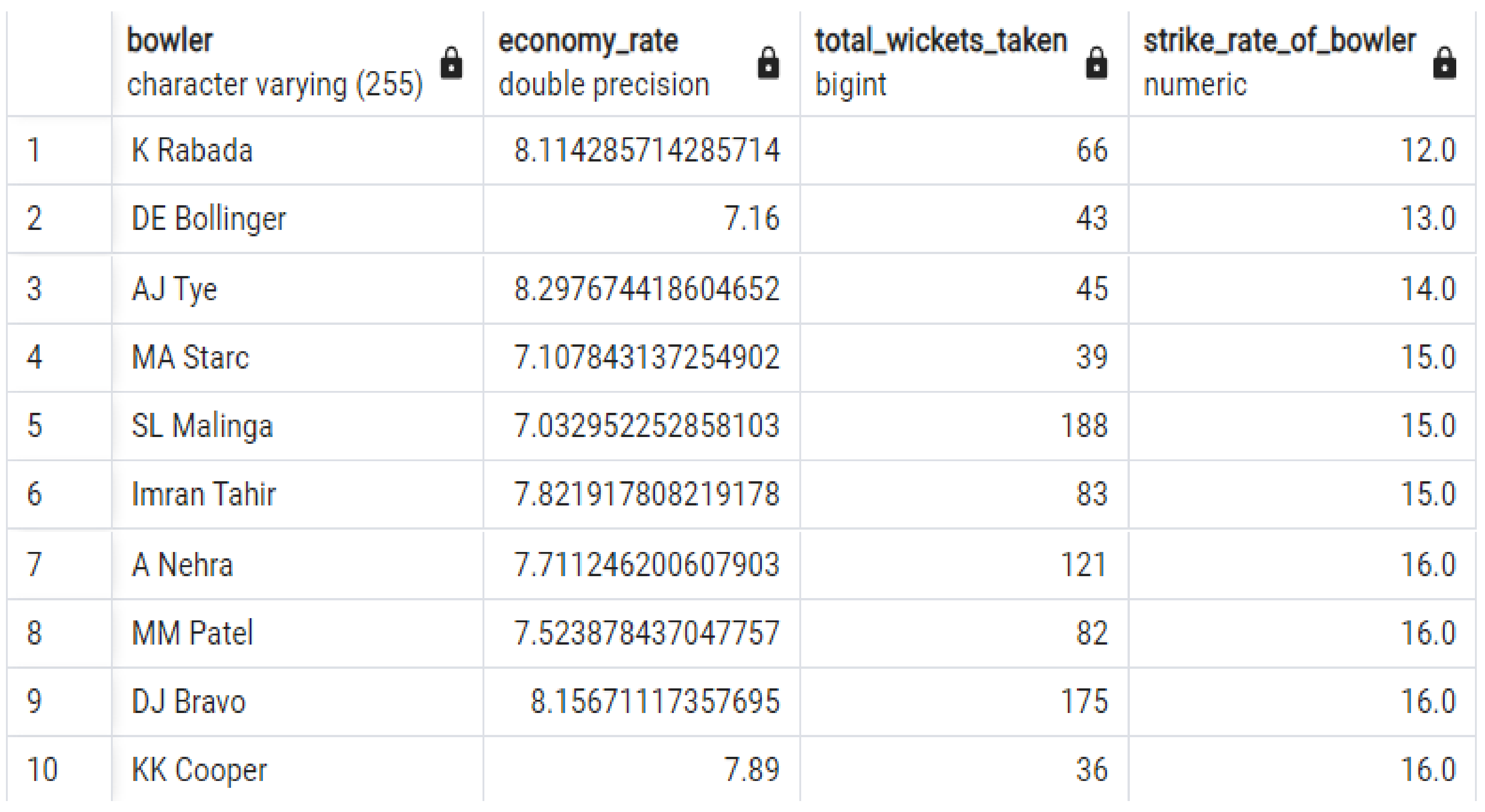


	bowler character varying (255)	economy_rate double precision
1	Rashid Khan	6.334228187919463
2	A Kumble	6.646998982706002
3	M Muralitharan	6.677235256816741
4	DW Steyn	6.769771528998243
5	R Ashwin	6.7736699729486025
6	SP Narine	6.815864022662889
7	DL Vettori	6.83312101910828
8	Washington Sundar	6.890909090909091
9	J Botha	6.922425952045134
10	R Tewatia	6.991482112436116

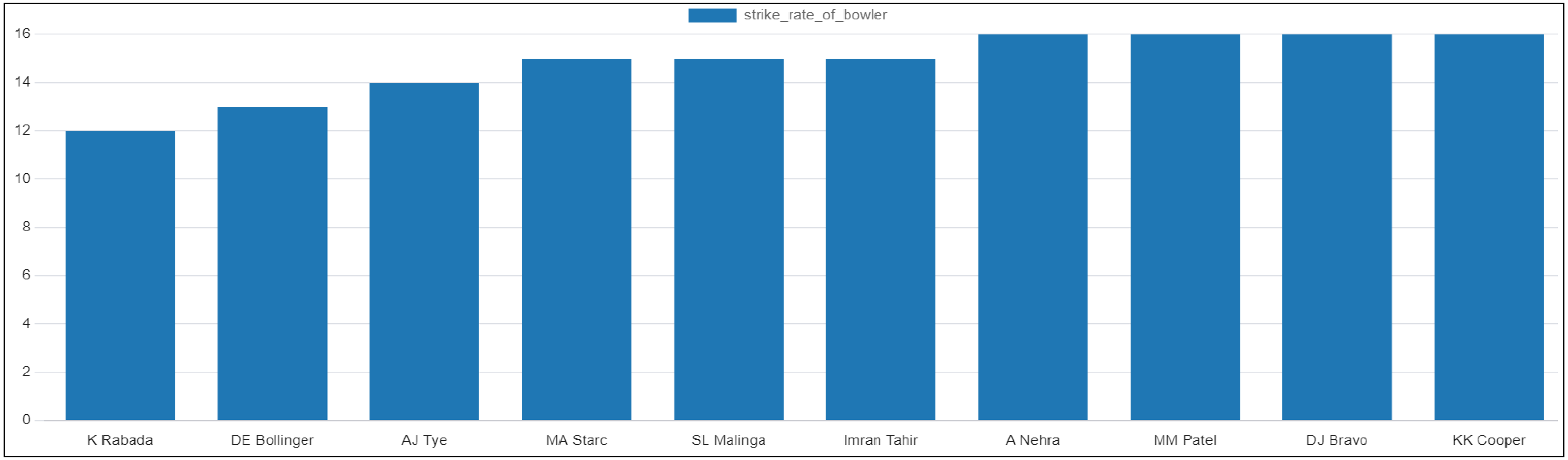


TASK 5
(Finding the bowlers having good bowling strike rate)

```
select bowler, (total_runs/(total_deliveries/6.0)) as economy_rate,
                    total_wickets_taken,
                    (total_deliveries/total_wickets_taken*1.0) as
strike_rate_of_bowler
from (select bowler, sum(total_runs) as total_runs,
                         sum(case when is_wicket=1 then 1 else 0 end) as
total_wickets_taken,
                         count(ball) as total_deliveries from all_table group by
bowler)
where total_deliveries>=500
order by strike_rate_of_bowler limit 10
```

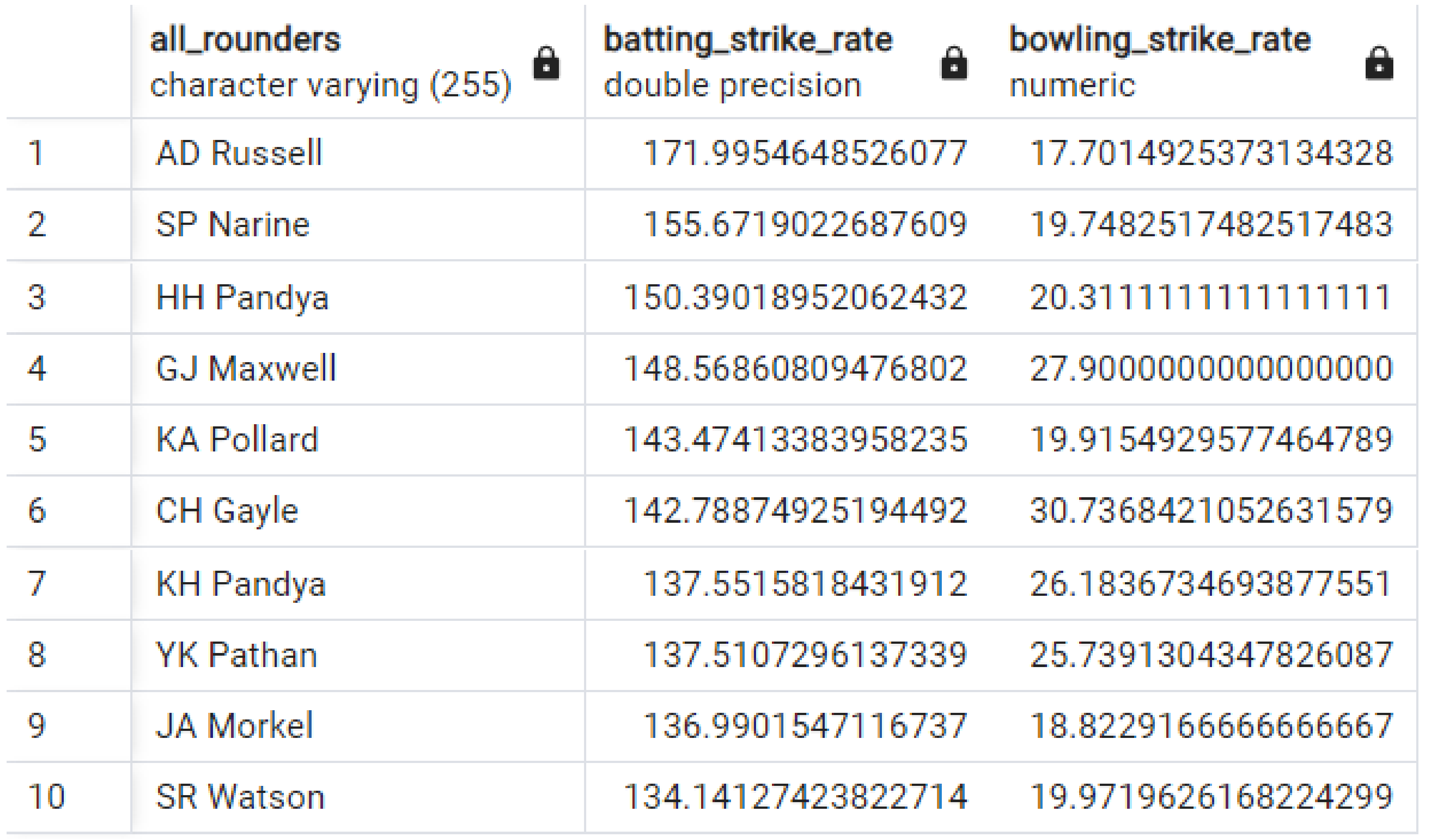


	bowler character varying (255)	economy_rate double precision	total_wickets_taken bigint	strike_rate_of_bowler numeric
1	K Rabada	8.114285714285714	66	12.0
2	DE Bollinger	7.16	43	13.0
3	AJ Tye	8.297674418604652	45	14.0
4	MA Starc	7.107843137254902	39	15.0
5	SL Malinga	7.032952252858103	188	15.0
6	Imran Tahir	7.821917808219178	83	15.0
7	A Nehra	7.711246200607903	121	16.0
8	MM Patel	7.523878437047757	82	16.0
9	DJ Bravo	8.15671117357695	175	16.0
10	KK Cooper	7.89	36	16.0

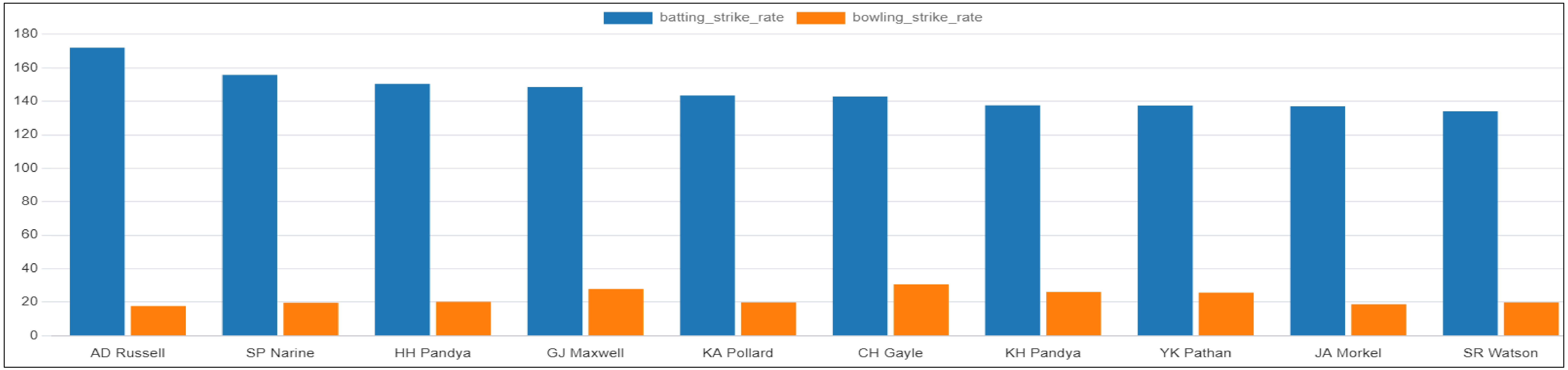


TASK 6
(Find the list of all rounders)

```
-- Creating table for batters and their strike rates
create table batters as
select batsman, sum(batsman_runs)/count(ball)*100 as batting_strike_rate
from all_table group by batsman having count(ball)>500
order by batting_strike_rate

-- Creating table for bowlers and their strike rates
create table bowlers as
select bowler, (count(ball)*1.0)/(sum(case when is_wicket=1 then 1 else 0 end)*1.0)
as bowling_strike_rate
from all_table group by bowler having count(ball)>300
order by bowler

-- Selecting the all rounders from batters and bowlers
select batters.batsman as all_rounders, batters.batting_strike_rate,
          bowlers.bowling_strike_rate
from batters join bowlers
on batters.batsman=bowlers.bowler  -- This will help us to find the common names
who can do batting and bowling as well
order by batting_strike_rate desc, bowling_strike_rate desc limit 10

-- After selecting the table dropping the temporary tables of batters and bowlers
drop table batters, bowlers
```

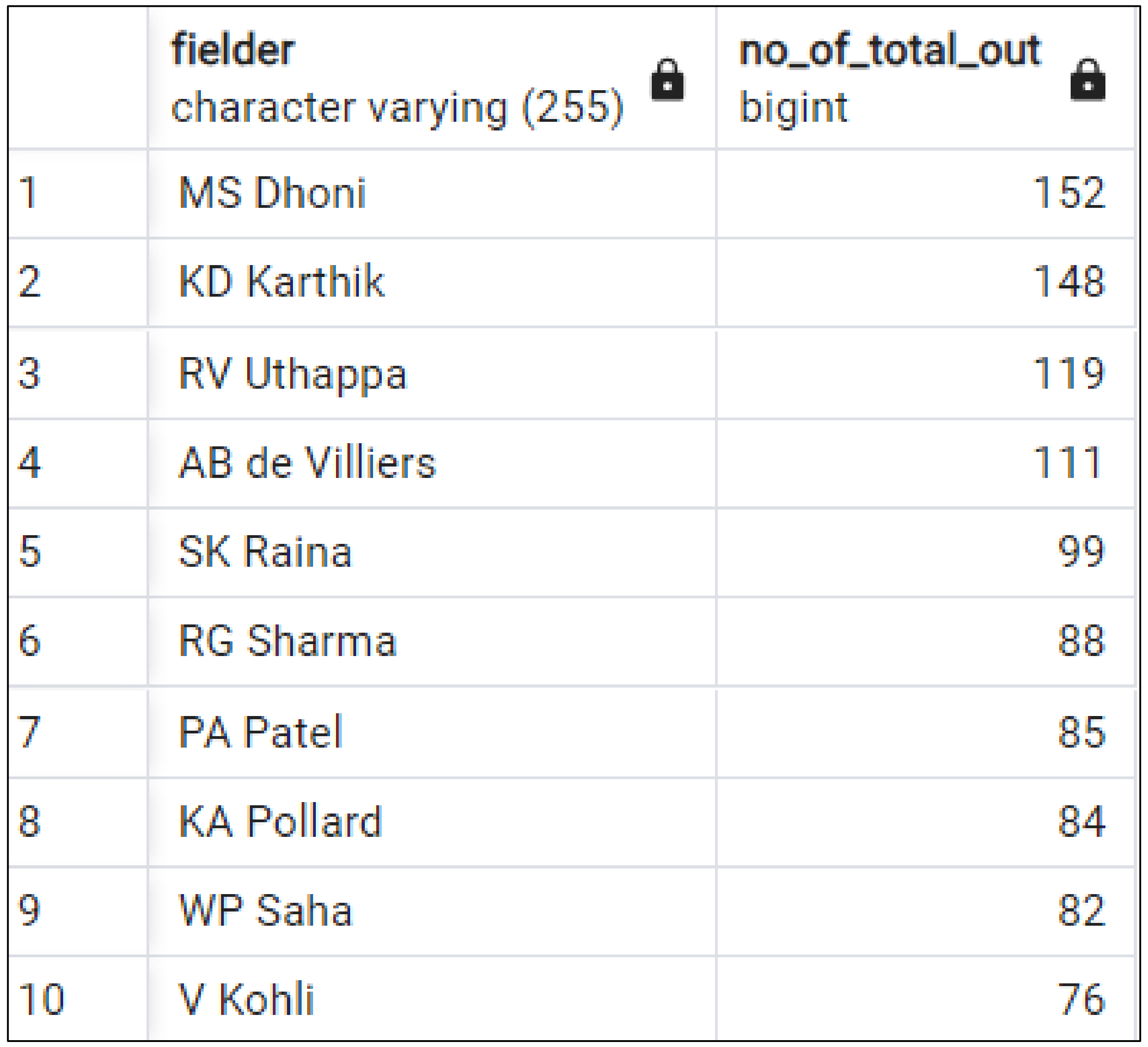


	all_rounders character varying (255)	batting_strike_rate double precision	bowling_strike_rate numeric
1	AD Russell	171.9954648526077	17.7014925373134328
2	SP Narine	155.6719022687609	19.7482517482517483
3	HH Pandya	150.39018952062432	20.3111111111111111
4	GJ Maxwell	148.56860809476802	27.9000000000000000
5	KA Pollard	143.47413383958235	19.9154929577464789
6	CH Gayle	142.78874925194492	30.7368421052631579
7	KH Pandya	137.5515818431912	26.1836734693877551
8	YK Pathan	137.5107296137339	25.7391304347826087
9	JA Morkel	136.9901547116737	18.8229166666666667
10	SR Watson	134.14127423822714	19.9719626168224299

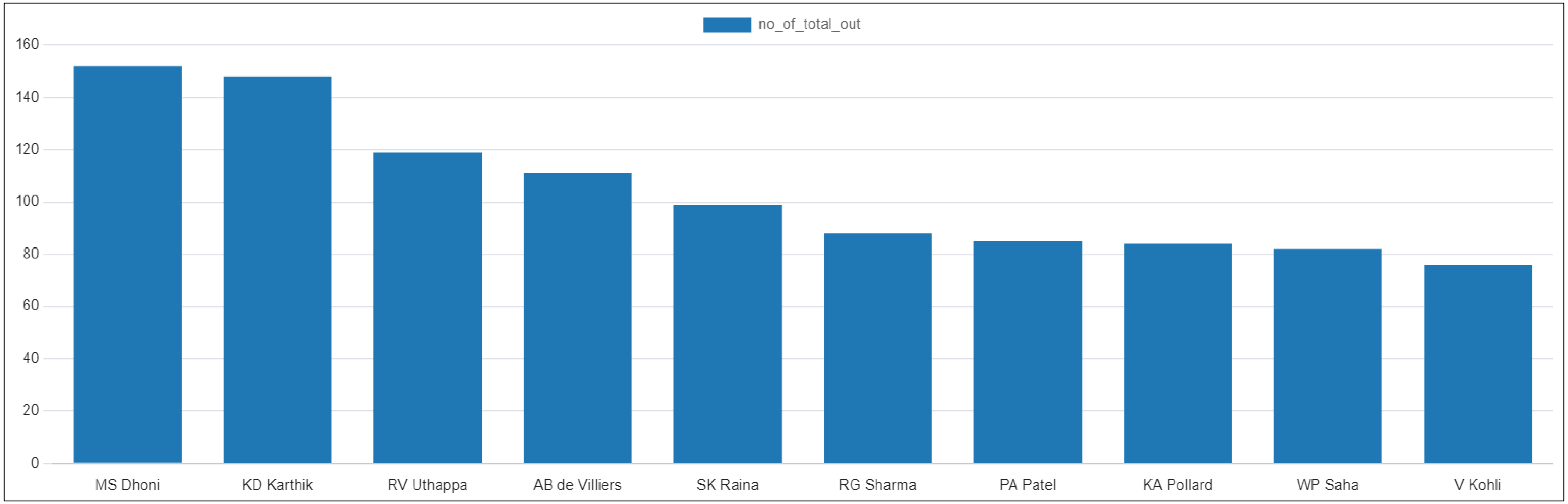


TASK 7
(Finding the list of wicketkeepers)

```
-- Creating table for fielders with their various kind of out with count
create table fielders_name as
select fielder, sum(case when dismissal_kind='caught' then 1 else 0 end) as no_catch,
                sum(case when dismissal_kind='run_out' then 1 else 0 end) as no_run_out,
                sum(case when dismissal_kind='stumped' then 1 else 0 end) as no_stumped
from all_table group by fielder

select * from fielders_name


-- Creating table to get the total number of out by each fielders
create table fileder_with_outs as
select *, (no_catch+no_run_out+no_stumped) as no_of_total_out from fielders_name order by no_of_total_out desc

select * from fileder_with_outs


-- Findings the wicket keeper
select fielder, no_of_total_out from (select all_table.dismissal_kind, fileder_with_outs.*
            from all_table join fileder_with_outs
            on all_table.fielder=fileder_with_outs.fielder
            order by no_of_total_out desc)
where dismissal_kind in ('caught', 'run_out', 'stumped')
AND
fielder not in ('NA') group by fielder, no_of_total_out
order by no_of_total_out desc limit 10

drop table fielders_name, fileder_with_outs
```



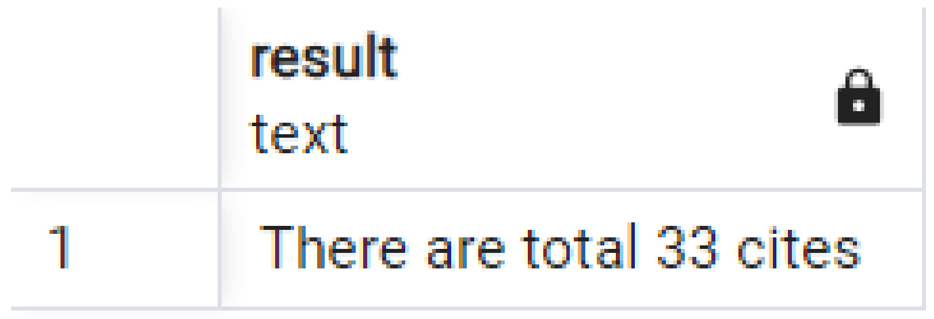
	fielder character varying (255)	no_of_total_out bigint
1	MS Dhoni	152
2	KD Karthik	148
3	RV Uthappa	119
4	AB de Villiers	111
5	SK Raina	99
6	RG Sharma	88
7	PA Patel	85
8	KA Pollard	84
9	WP Saha	82
10	V Kohli	76



<u>SOME ADDITIONAL QUESTION FOR FINAL ASSESSMENT</u>

Q1 Get the count of cities that have hosted an IPL match

```
select
'There are total '|| (select count(distinct city) from all_table) || ' cites' as result
```



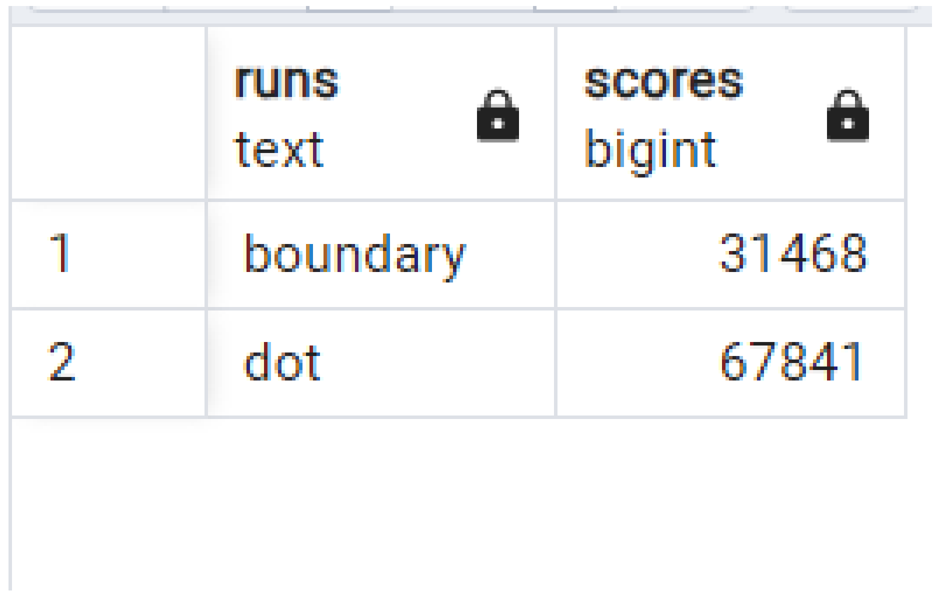
```
   result
   text
1  There are total 33 cites
```

Q2 Create table deliveries_v02 with all the columns of the table 'deliveries' and an additional column ball_result containing values boundary, dot or other depending on the total_run

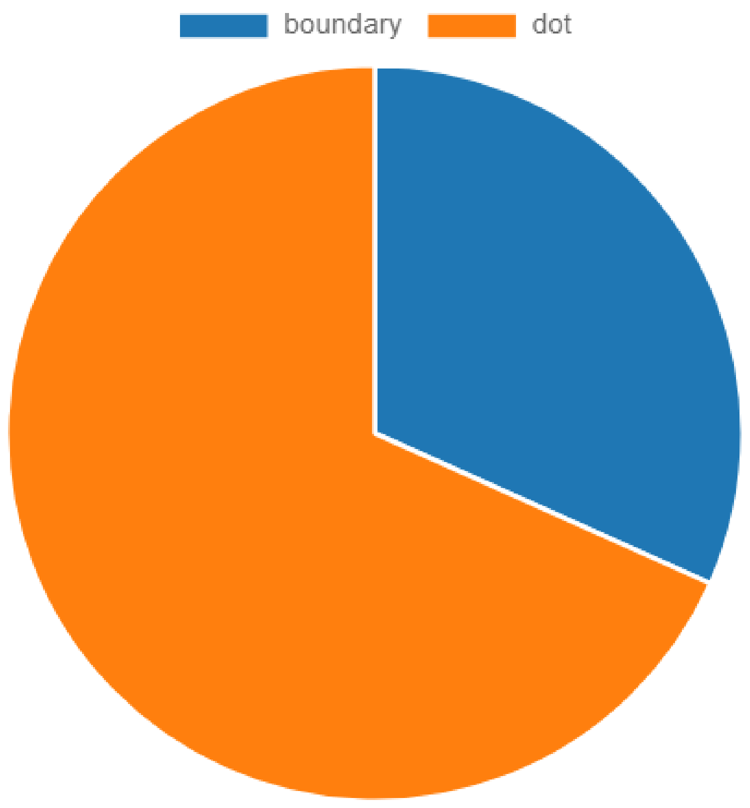
```
create table deliveries_v02 as
select *, case
                    when total_runs>=4 then 'boundary'
                    when total_runs=0 then 'dot'
                    else 'other'
                    end as runs
from ipl_ball

select * from deliveries_v02
```

Q3 Write a query to fetch the total number of boundaries and dot balls from the deliveries_v02 table

```
select runs, count(runs) as scores from deliveries_v02
where runs in ('boundary', 'dot')
group by runs
```

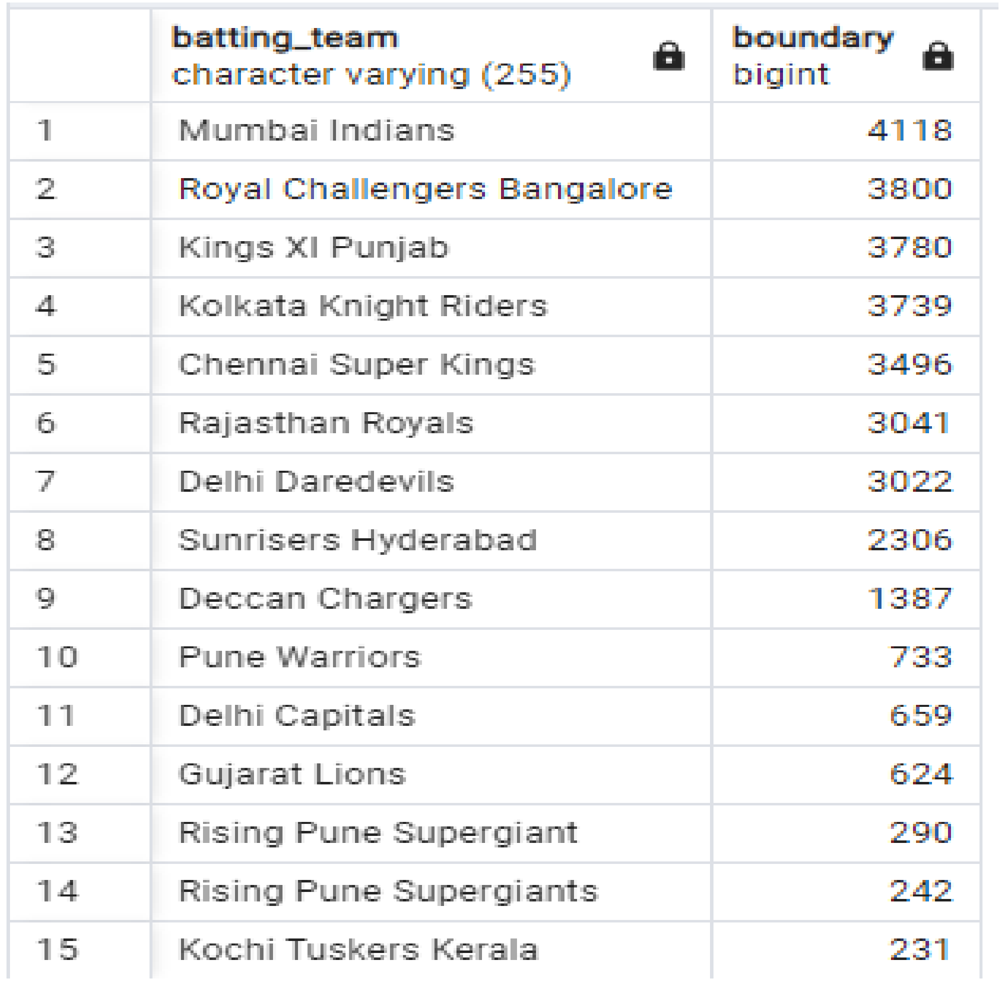


	runs text	scores bigint
1	boundary	31468
2	dot	67841

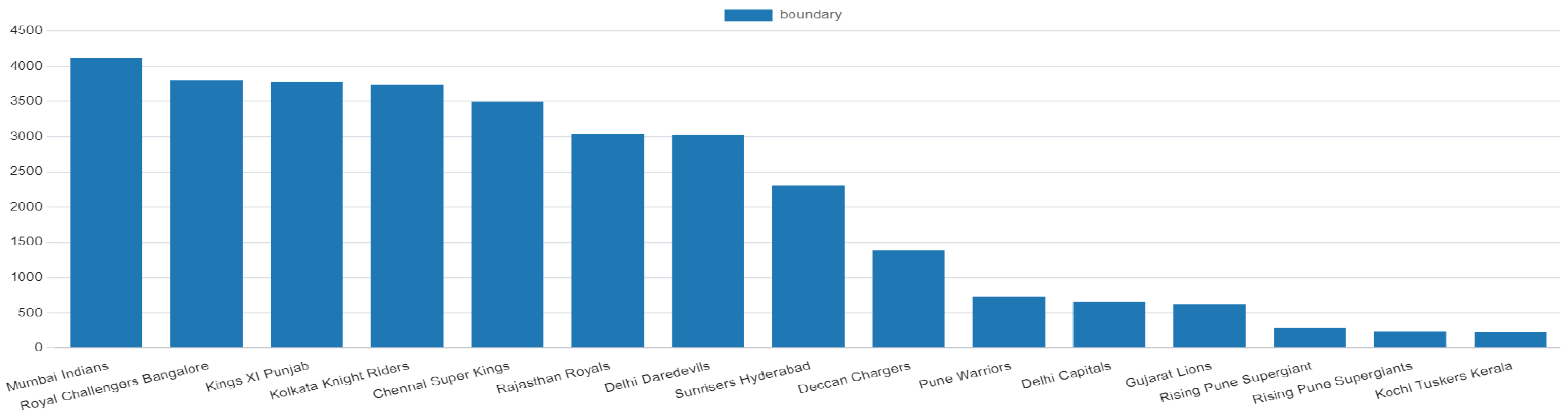


Q4 Write a query to fetch the total number of boundaries scored by each team from the deliveries_v02 table and order it in descending order of the number of boundaries scored

```
select batting_team, count(runs) as boundary
        from deliveries_v02
        where runs='boundary'
        group by batting_team order by boundary desc
```

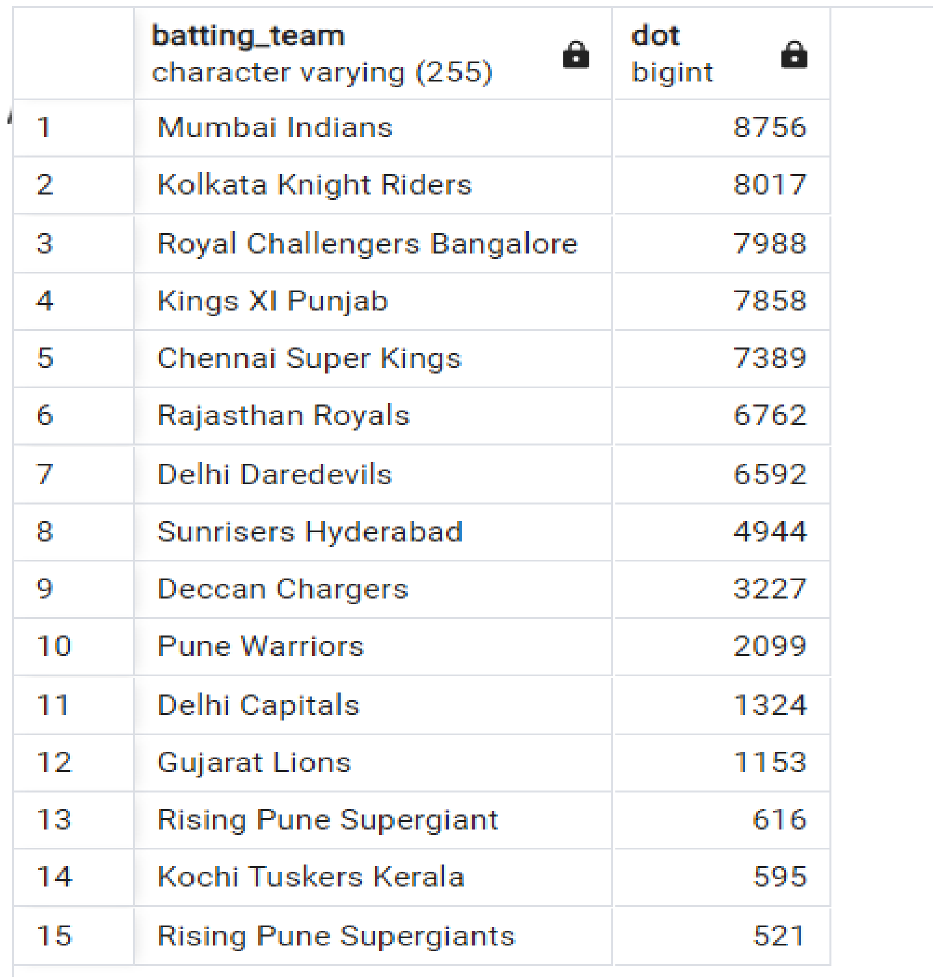


	batting_team character varying (255)	boundary bigint
1	Mumbai Indians	4118
2	Royal Challengers Bangalore	3800
3	Kings XI Punjab	3780
4	Kolkata Knight Riders	3739
5	Chennai Super Kings	3496
6	Rajasthan Royals	3041
7	Delhi Daredevils	3022
8	Sunrisers Hyderabad	2306
9	Deccan Chargers	1387
10	Pune Warriors	733
11	Delhi Capitals	659
12	Gujarat Lions	624
13	Rising Pune Supergiant	290
14	Rising Pune Supergiants	242
15	Kochi Tuskers Kerala	231

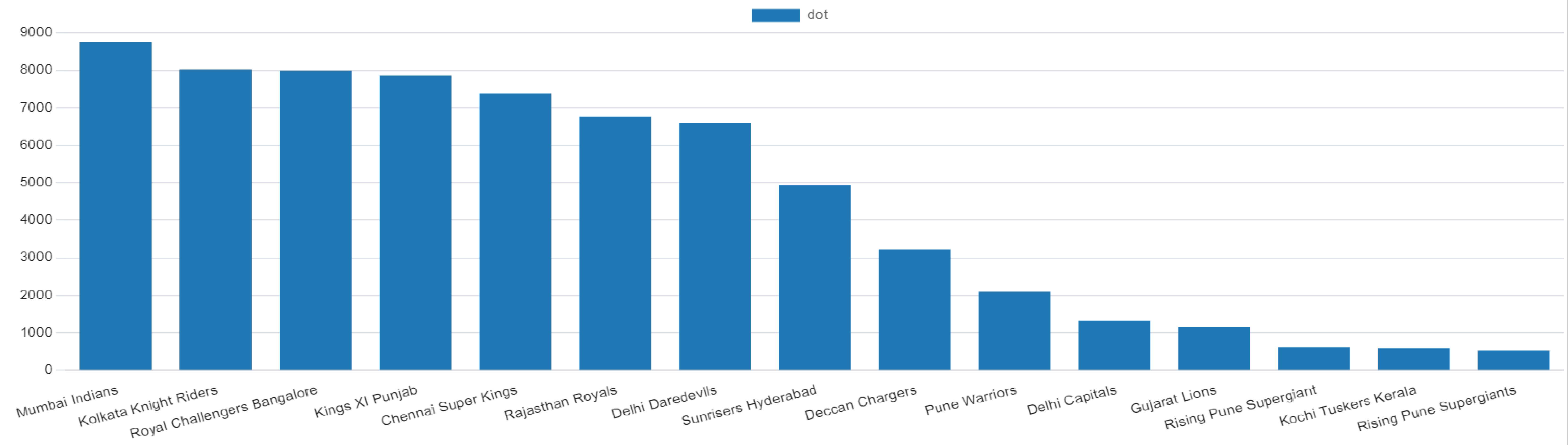


Q5 Write a query to fetch the total number of dot balls bowled by each team and order it in descending order of the total number of dot balls bowled

```
select batting_team, count(runs) as dot
        from deliveries_v02
        where runs='dot'
        group by batting_team order by dot desc
```

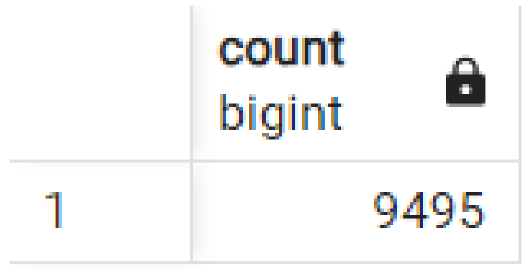


	batting_team character varying (255)	dot bigint
1	Mumbai Indians	8756
2	Kolkata Knight Riders	8017
3	Royal Challengers Bangalore	7988
4	Kings XI Punjab	7858
5	Chennai Super Kings	7389
6	Rajasthan Royals	6762
7	Delhi Daredevils	6592
8	Sunrisers Hyderabad	4944
9	Deccan Chargers	3227
10	Pune Warriors	2099
11	Delhi Capitals	1324
12	Gujarat Lions	1153
13	Rising Pune Supergiant	616
14	Kochi Tuskers Kerala	595
15	Rising Pune Supergiants	521



Q6 Write a query to fetch the total number of dismissals by dismissal kinds where dismissal kind is not NA

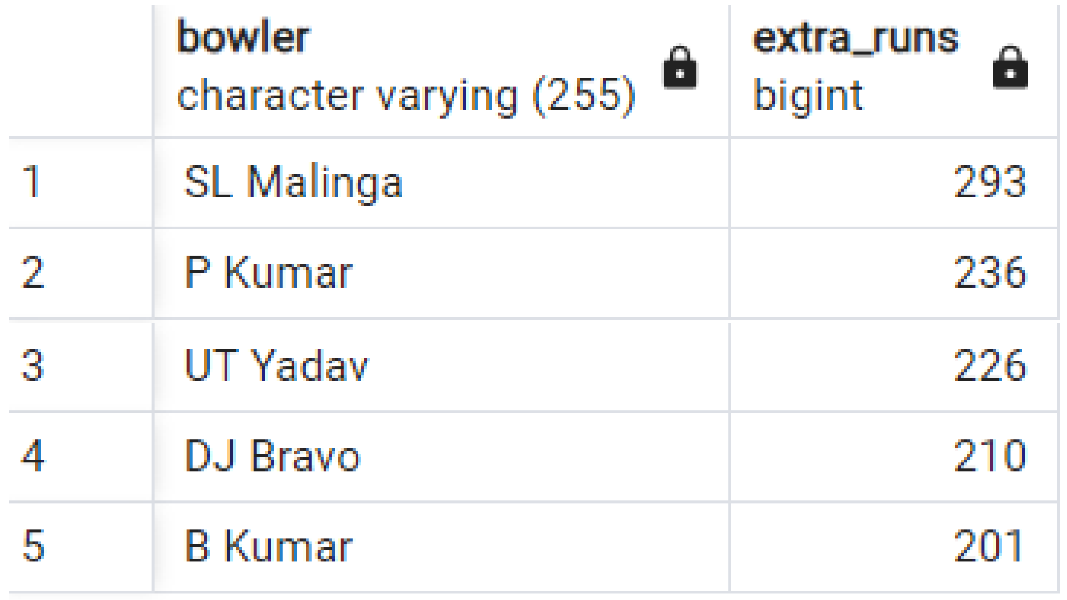
```
select count(dismissal_kind)
from deliveries_v02 where dismissal_kind!='NA'
```



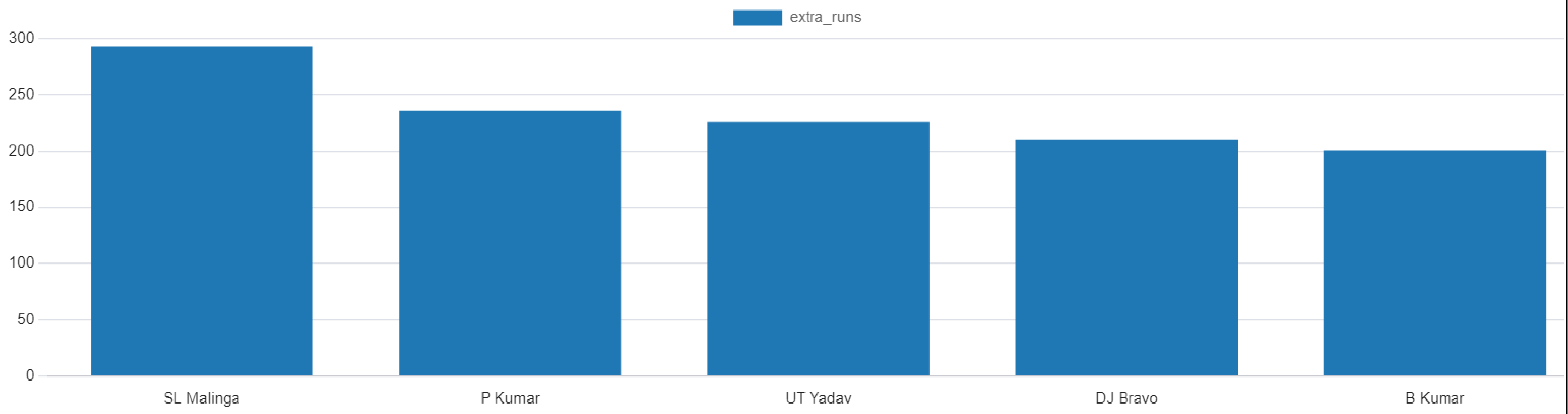
	count bigint
1	9495

Q7 Write a query to get the top 5 bowlers who conceded maximum extra runs from the deliveries table

```
select bowler, sum(extra_runs) as extra_runs
from deliveries_v02
group by bowler order by extra_runs desc limit 5
```



	bowler character varying (255)	extra_runs bigint
1	SL Malinga	293
2	P Kumar	236
3	UT Yadav	226
4	DJ Bravo	210
5	B Kumar	201



Q8 Write a query to create a table named deliveries_v03 with all the columns of deliveries_v02 table and two additional column (named venue and match_date) of venue and date from table matches

```
create table deliveries_v03 as
select deliveries_v02.*, ipl_matches.venue, ipl_matches.match_date
from deliveries_v02 join ipl_matches
on deliveries_v02.id=ipl_matches.id

select * from deliveries_v03
```

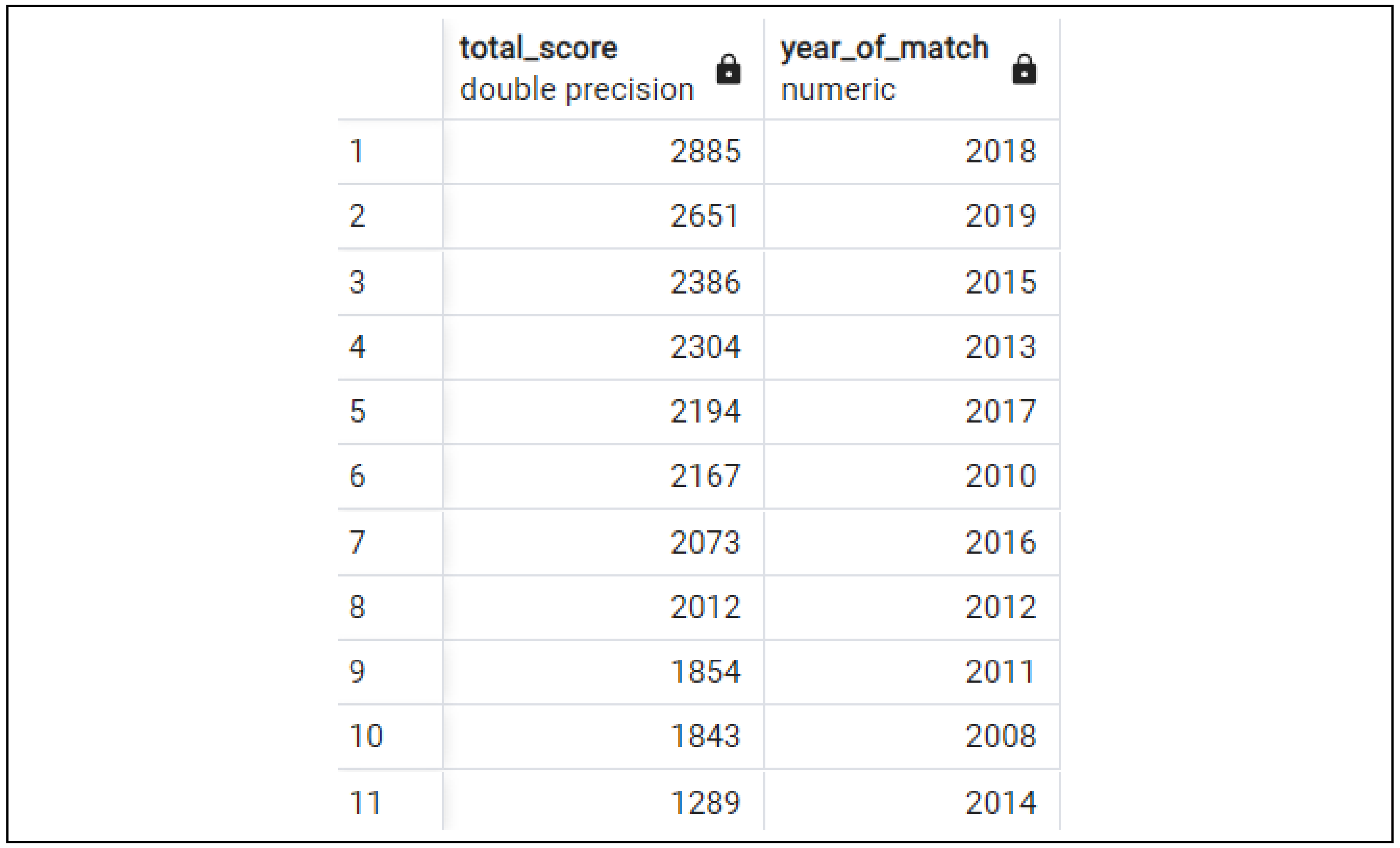
Q9 Write a query to fetch the total runs scored for each venue and order it in the descending order of total runs scored

```
select venue, sum(total_runs) as total_score from deliveries_v03
 group by venue order by total_score desc
```

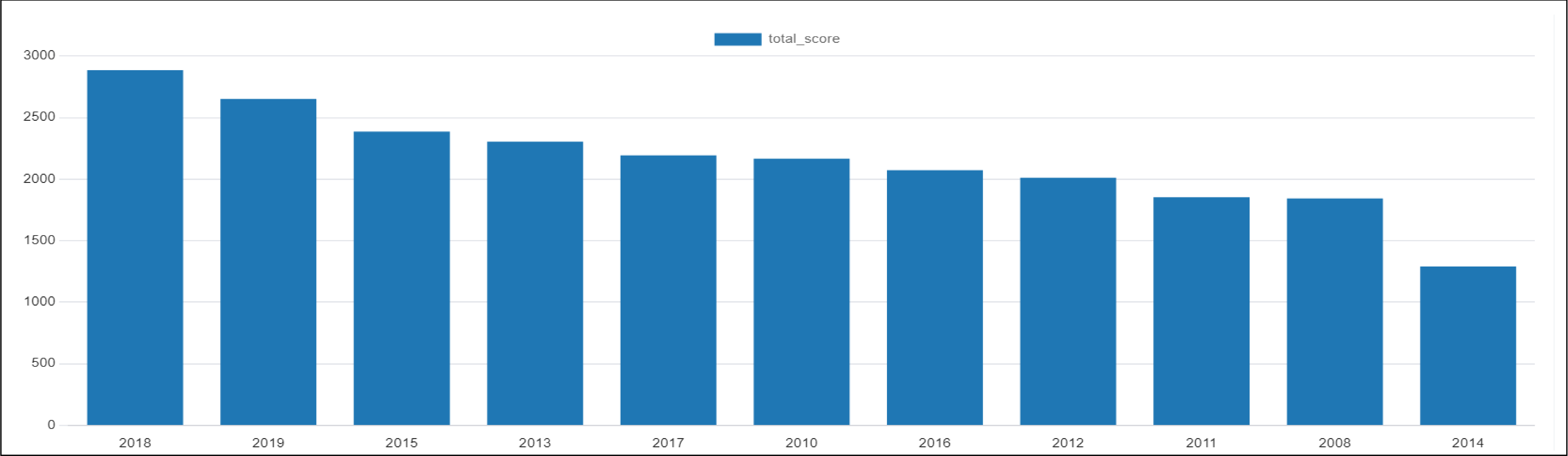
venue	total_score
Eden Gardens	23658
Wankhede Stadium	23390
Feroz Shah Kotla	22947
M Chinnaswamy Stadium	20237
Rajiv Gandhi International Stadium, Uppal	19484
MA Chidambaram Stadium, Chepauk	17821
Sawai Mansingh Stadium	14264
Punjab Cricket Association Stadium, Mohali	10987
Dubai International Cricket Stadium	10402
Sheikh Zayed Stadium	8830
Punjab Cricket Association IS Bindra Stadium, Mohali	7021
Maharashtra Cricket Association Stadium	6780
Sharjah Cricket Stadium	5924
M.Chinnaswamy Stadium	5127
Dr DY Patil Sports Academy	4810
Subrata Roy Sahara Stadium	4755
Kingsmead	4353
Brabourne Stadium	3842
Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket Stadium	3746
Sardar Patel Stadium, Motera	3746
SuperSport Park	3653
Saurashtra Cricket Association Stadium	3316
Himachal Pradesh Cricket Association Stadium	2897
Holkar Cricket Stadium	2872
New Wanderers Stadium	2292
Barabati Stadium	2278
JSCA International Stadium Complex	2056
St George's Park	2033
Newlands	1764
Shaheed Veer Narayan Singh International Stadium	1741
Nehru Stadium	1363
Green Park	1298
De Beers Diamond Oval	897
Vidarbha Cricket Association Stadium, Jamtha	882
Buffalo Park	799
OUTsurance Oval	529

Q10 Write a query to fetch the year-wise total runs scored at Eden Gardens and order it in the descending order of total runs scored

```
select sum(total_runs) as total_score,
          extract(year from match_date) as year_of_match
from all_table
group by venue, extract(year from match_date)
having venue='Eden Gardens'
order by total_score desc
```



	total_score double precision	year_of_match numeric
1	2885	2018
2	2651	2019
3	2386	2015
4	2304	2013
5	2194	2017
6	2167	2010
7	2073	2016
8	2012	2012
9	1854	2011
10	1843	2008
11	1289	2014


total_score