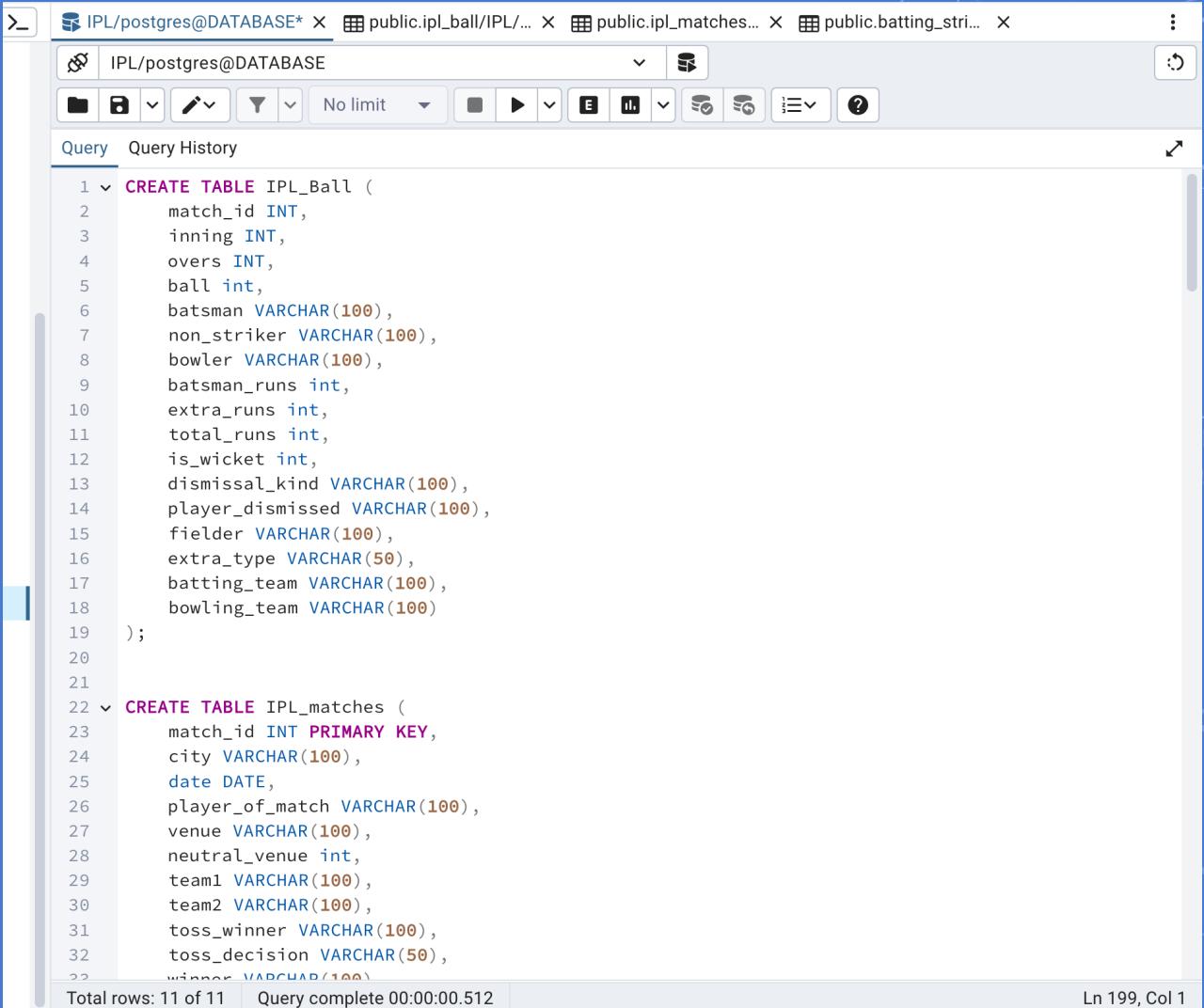


SQL ASSIGNMENT

SUMIT UPADHYAY

1ST I HAVE CREATED
THE DB AND
INSERTED THE CSV
DATA IN IT



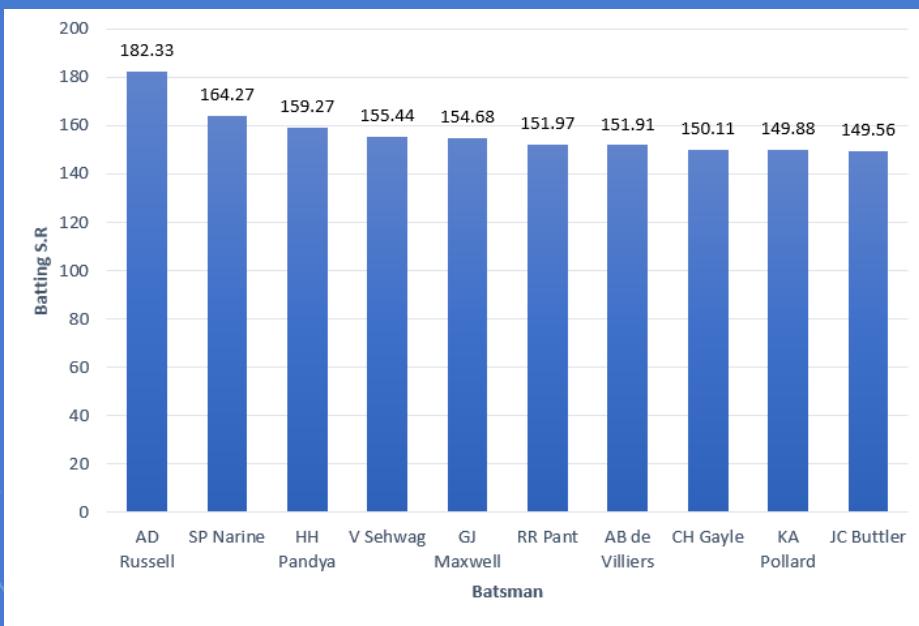
The screenshot shows a PostgreSQL client interface with two CREATE TABLE statements in the query editor:

```
1 v CREATE TABLE IPL_Ball (
2   match_id INT,
3   inning INT,
4   overs INT,
5   ball int,
6   batsman VARCHAR(100),
7   non_striker VARCHAR(100),
8   bowler VARCHAR(100),
9   batsman_runs int,
10  extra_runs int,
11  total_runs int,
12  is_wicket int,
13  dismissal_kind VARCHAR(100),
14  player_dismissed VARCHAR(100),
15  fielder VARCHAR(100),
16  extra_type VARCHAR(50),
17  batting_team VARCHAR(100),
18  bowling_team VARCHAR(100)
19 );
20
21
22 v CREATE TABLE IPL_matches (
23   match_id INT PRIMARY KEY,
24   city VARCHAR(100),
25   date DATE,
26   player_of_match VARCHAR(100),
27   venue VARCHAR(100),
28   neutral_venue int,
29   team1 VARCHAR(100),
30   team2 VARCHAR(100),
31   toss_winner VARCHAR(100),
32   toss_decision VARCHAR(50),
33   winner VARCHAR(100)
34 );
```

Total rows: 11 of 11 Query complete 00:00:00.512 Ln 199, Col 1

YOUR PRIORITY IS TO GET 2-3 PLAYERS WITH HIGH S.R WHO HAVE FACED AT LEAST 500 BALLS. AND TO DO THAT YOU HAVE TO MAKE A LIST OF 10 PLAYERS YOU WANT TO BID IN THE AUCTION SO THAT WHEN YOU TRY TO GRAB THEM IN AUCTION YOU SHOULD NOT PAY THE AMOUNT GREATER THAN YOU HAVE IN THE PURSE FOR A PARTICULAR PLAYER.

(STRIKE RATE IS TOTAL RUNS SCORED BY BATSMAN DIVIDED BY NUMBER OF BALLS FACED BUT REMEMBER WHEN EXTRAS TYPE IS 'WIDES' IT IS NOT COUNTED AS A BALL FACED NEITHER COUNTED AS BATSMEN RUNS)



Object Explorer IPL/postgres@DATABASE* public.ipl_ball/IPL... public.ipl_matches...

Query Query History

```

33   winner VARCHAR(100),
34   results VARCHAR(50),
35   result_margin INT,
36   eliminator varchar(50),
37   methods varchar(50),
38   umpire1 VARCHAR(100),
39   umpire2 VARCHAR(100)
40 );
41
42 <Select batsman, sum(batsman_runs) as runs_scored,
43 Count(batsman_runs) as balls_faced,
44 Sum(batsman_runs*100)*1.0/count(batsman_runs) as batting_strike_rate
45 From ipl_ball where not extra_type = 'wides' group by batsman
46 Having count(batsman_runs)>500 order by batting_strike_rate desc limit 10;
47

```

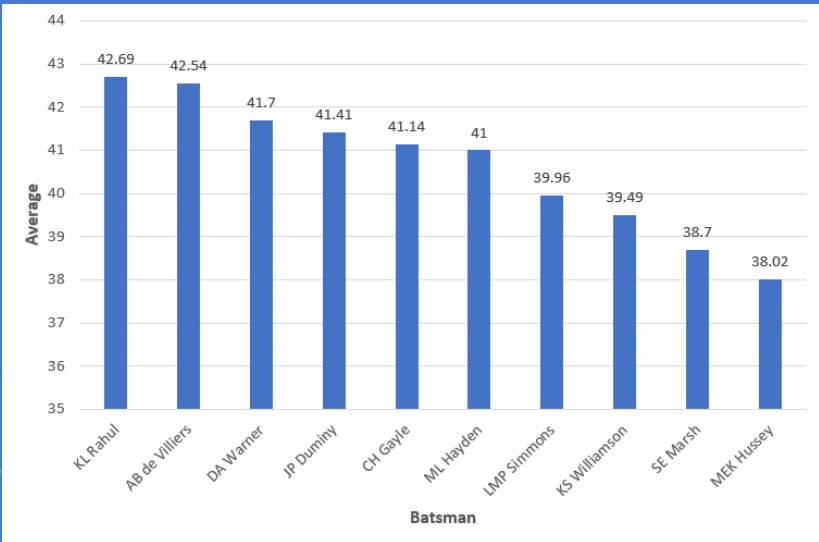
Data Output Messages Notifications

batsman	runs_scored	balls_faced	batting_strike_rate
1 AD Russell	1517	832	182.3317307692307692
2 SP Narine	892	543	164.2725598526703499
3 HH Pandya	1349	847	159.2680047225501771
4 V Sehwag	2728	1755	155.4415954415954416
5 GJ Maxwell	1505	973	154.67625899205263158
6 RR Pant	2079	1368	151.9736842105263158
7 AB de Villiers	4849	3192	151.9110275689223058
8 CH Gayle	4772	3179	150.1100975149418056
9 KA Pollard	3023	2017	149.8760535448686168
10 JC Buttler	1714	1146	149.5636998254799302

Total rows: 10 of 10 Query complete 00:00:00.373

Ln 47, Col 1

NOW YOU NEED TO GET 2-3 PLAYERS WITH GOOD AVERAGE WHO HAVE PLAYED MORE THAN 2 IPL SEASONS. AND TO DO THAT YOU HAVE TO MAKE A LIST OF 10 PLAYERS YOU WANT TO BID IN THE AUCTION SO THAT WHEN YOU TRY TO GRAB THEM IN AUCTION YOU SHOULD NOT PAY THE AMOUNT GREATER THAN YOU HAVE IN THE PURSE FOR A PARTICULAR PLAYER.



Object Explorer IPL/postgres@DATABASE* public.ipl_ball/IPL... public.ipl_matches...

Query Query History

```

41
42 v Select batsman, sum(batsman_runs) as runs_scored,
43 Count(batsman_runs) as balls_faced,
44 Sum(batsman_runs*100)*1.0/count(batsman_runs) as batting_strike_rate
45 From ipl_ball where not extra_type = 'wides' group by batsman
46 Having count(batsman_runs)>500 order by batting_strike_rate desc limit 10;
47
48
49 v Select batsman,sum(batsman_runs) as runs_scored,
50 Count(distinct match_id) as innings_batted,
51 Sum(is_wicket) as Number_of_times_dismissed,
52 sum (batsman_runs)*1.0/sum(is_wicket) as batting_average
53 From ipl_ball group by batsman having count(distinct match_id)> 28
54 order by batting_average desc limit 10;
55

```

Data Output Messages Notifications

batsman	runs_scored	innings_batted	number_of_times_dismissed	batting_average
1 KL Rahul	2647	72	62	42.6935483870967742
2 AB de Villiers	4849	156	114	42.5350877192982456
3 DA Warner	5254	142	126	41.6984126984126984
4 JP Duminy	2029	75	49	41.4081632653061224
5 CH Gayle	4772	131	116	41.1379310344827586
6 ML Hayden	1107	32	27	41.0000000000000000
7 LMP Simmons	1079	29	27	39.9629629629629630
8 KS Williamson	1619	52	41	39.4878048780487805
9 SE Marsh	2477	69	64	38.7031250000000000
10 MEK Hussey	1977	58	52	38.0192307692307692

Total rows: 10 of 10 Query complete 00:00:00.777

Ln 49, Col 1

NOW YOU NEED TO GET 2-3 HARD-HITTING PLAYERS WHO HAVE SCORED MOST RUNS IN BOUNDARIES AND HAVE PLAYED MORE THAN THE 2 IPL SEASONS. TO DO THAT YOU HAVE TO MAKE A LIST OF 10 PLAYERS YOU WANT TO BID IN THE AUCTION SO THAT WHEN YOU TRY TO GRAB THEM IN AUCTION YOU SHOULD NOT PAY THE AMOUNT GREATER THAN YOU HAVE IN THE PURSE FOR A PARTICULAR PLAYER.

1	BATSMAN	BOUNDARY_PERCENTAGE
2	SP Narine	81.17
3	AD Russell	78.71
4	CH Gayle	76.07
5	ST Jayasuriya	74.22
6	AC Gilchrist	72.89
7	V Sehwag	72.29
8	DR Smith	70.52
9	CA Lynn	69.53
10	Harbhajan Singh	68.52
11	SR Watson	68.25

The screenshot shows a PostgreSQL database interface with the following details:

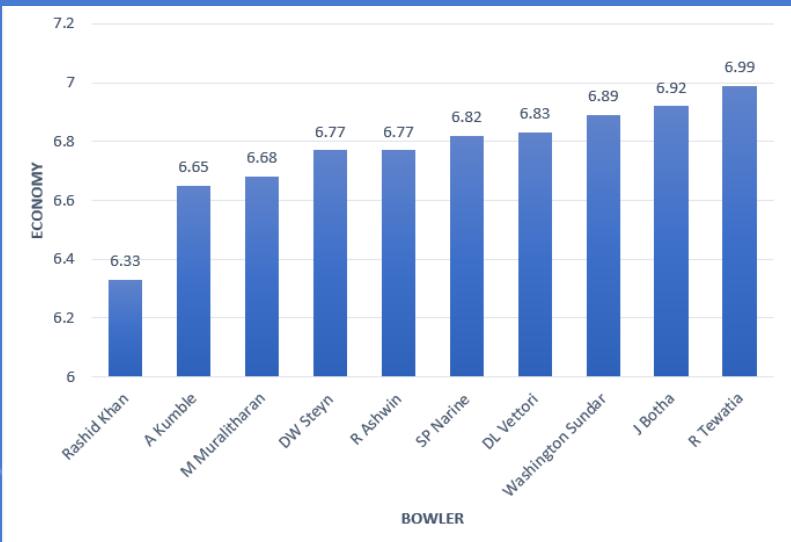
- Object Explorer:** Shows the schema structure under the "public" namespace, including Aggregates, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, and Tables (10).
- Query Editor:** Displays the SQL query used to generate the results:


```

SELECT
  batsman,
  ROUND(SUM(CASE WHEN batsman_runs IN(4,6)
                  THEN batsman_runs ELSE 0 END)*1.0 / SUM(batsman_runs)*100,2) AS boundary_percentage
FROM ipl_ball
WHERE
  extra_type NOT IN ('wides')
GROUP BY
  batsman
HAVING
  COUNT(DISTINCT match_id) > 28
ORDER BY
  boundary_percentage DESC
LIMIT 10;
      
```
- Data Output:** Shows the top 10 results from the query:

	batsman	boundary_percentage
1	SP Narine	81.17
2	AD Russell	78.71
3	CH Gayle	76.07
4	ST Jayasuriya	74.22
5	AC Gilchrist	72.89
6	V Sehwag	72.29
7	DR Smith	70.52
8	CA Lynn	69.53
9	Harbhajan Singh	68.52
10	SR Watson	68.25
- Message Bar:** Shows "Total rows: 10 of 10" and "Query complete 00:00:00.546".
- Bottom Right:** A small note "Ln 73, Col 1".

YOUR PRIORITY IS TO GET 2-3 BOWLERS WITH GOOD ECONOMY WHO HAVE BOWLED AT LEAST 500 BALLS IN IPL SO FAR. TO DO THAT YOU HAVE TO MAKE A LIST OF 10 PLAYERS YOU WANT TO BID IN THE AUCTION SO THAT WHEN YOU TRY TO GRAB THEM IN AUCTION YOU SHOULD NOT PAY THE AMOUNT GREATER THAN YOU HAVE IN THE PURSE FOR A PARTICULAR PLAYER.



Object Explorer IPL/postgres@DATABASE* public.ipl_ball/IPL/... public.ipl_matches...

```

80 v create table Bowling_economy
81   as select a.bowler,
82             a.runs_got,
83             b.balls_bowled
84   from B_strike_rate as a left join balls
85     as b on a.bowler=b.bowler order by bowler;
86
87 select*from Bowling_economy;
88
89
90 v Select bowler,balls_bowled,runs_got,
91   balls_bowled/6 as over, (runs_got*6)*1.0/(balls_bowled)
92   as economy from bowling_economy where balls_bowled > 500
93   order by economy asc limit 10 ;

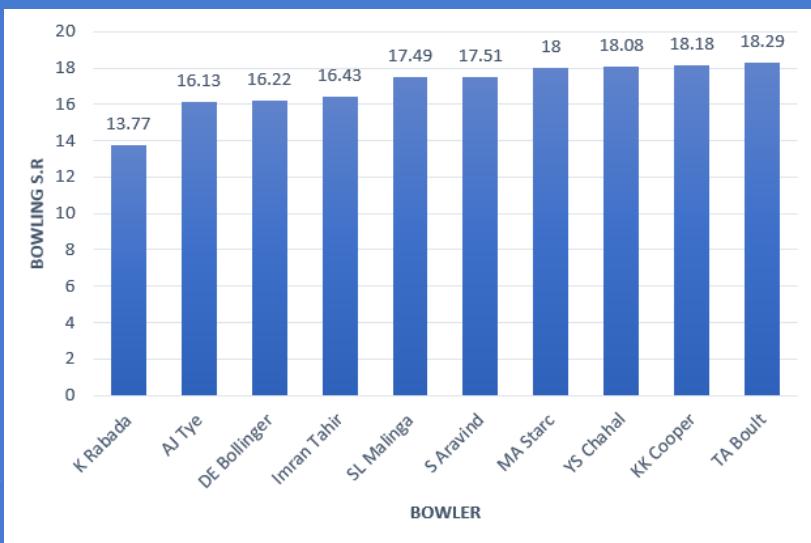
```

Data Output Messages Notifications

bowler	balls_bowled	runs_got	over	economy
Rashid Khan	1476	1542	246	6.2682926829268293
A Kumble	965	1058	160	6.5782383419689119
M Muralitharan	1524	1703	254	6.7047244094498189
DL Vettori	777	880	129	6.7953667953667954
SP Narine	2785	3158	464	6.803596642728905
Washington Sundar	654	749	109	6.8715596330275229
R Ashwin	3230	3709	538	6.8897832817337461
DW Steyn	2176	2514	362	6.9319852941176471
J Botha	694	806	115	6.9682997118155620
R Sharma	928	1089	154	7.0409482758620690

Total rows: 10 of 10 Query complete 00:00:00.233 Ln 89, Col 1

NOW YOU NEED TO GET 2-3 BOWLERS WITH THE BEST STRIKE RATE AND WHO HAVE BOWLED AT LEAST 500 BALLS IN IPL SO FAR. TO DO THAT YOU HAVE TO MAKE A LIST OF 10 PLAYERS YOU WANT TO BID IN THE AUCTION SO THAT WHEN YOU TRY TO GRAB THEM IN AUCTION YOU SHOULD NOT PAY THE AMOUNT GREATER THAN YOU HAVE IN THE PURSE FOR A PARTICULAR PLAYER.



Object Explorer IPL/postgres@DATABASE* public.ipl_ball/IPL... public.ipl_matches...

```

Object Explorer          IPL/postgres@DATABASE*          public.ipl_ball/IPL...          public.ipl_matches...
Servers (1)           DATABASE          No limit
  Databases (4)
    Classroom
    IPL
      Casts
      Catalogs
      Event Triggers
      Extensions
      Foreign Data Wrappers
      Languages
      Publications
    Schemas (1)
    public
      Aggregates
      Collations
      Domains
      FTS Configurations
      FTS Dictionaries
      FTS Parsers
      FTS Templates
      Foreign Tables
      Functions
      Materialized Views
      Operators
      Procedures
      Sequences
    Tables (2)
      ipl_ball
      ipl_matches
    Trigger Functions
    Types
  
```

Query Query History

```

order by economy asc limit 10;

Create table bowling_strikerate as select bowler, count(ball) as balls_bowled, count(case when dismissal_kind='caught' or dismissal_kind='bowled' or dismissal_kind='caught and bowled' or dismissal_kind='lbw' or dismissal_kind='hit wicket' or dismissal_kind='stumped' then 1 else null end) as total_wickets from ipl_ball where not extra_type='wides' and not extra_type='noballs' and not extra_type='penalty' group by bowler;
select *, balls_bowled*1.0/nullif(total_wickets,0) as bowling_strike_rate from bowling_strikerate where balls_bowled > 500
order by bowling_strike_rate asc limit 10;

```

Data Output Messages Notifications

bowler	balls_bowled	total_wickets	bowling_strike_rate
K Rabada	805	61	13.1967213114754098
AJ Tye	619	40	15.4750000000000000
DE Bollinger	576	36	16.0000000000000000
S Aravind	760	43	17.6744186046511628
Imran Tahir	1292	73	17.6986301369863014
SL Malinga	2827	159	17.7798742138364780
MA Starc	580	32	18.1250000000000000
Mohammed Siraj	716	39	18.3589743589743590
NM Coulter-Nile	719	39	18.4358974358974359
DJ Bravo	2712	147	18.4489795918367347

Total rows: 10 of 10 Query complete 00:00:00.122 Ln 104, Col 1

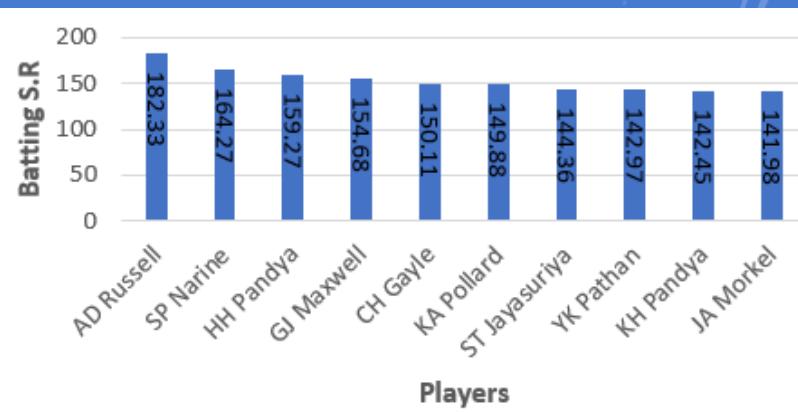
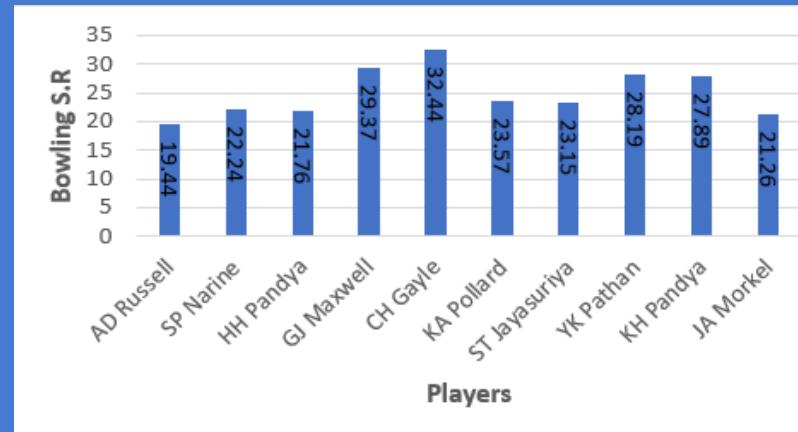
NOW YOU NEED TO GET 2-3 ALL_ROUNDERS WITH THE BEST BATTING AS WELL AS BOWLING STRIKE RATE AND WHO HAVE FACED AT LEAST 500 BALLS IN IPL SO FAR AND HAVE BOWLED MINIMUM 300 BALLS.TO DO THAT YOU HAVE TO MAKE A LIST OF 10 PLAYERS YOU WANT TO BID IN THE AUCTION SO THAT WHEN YOU TRY TO GRAB THEM IN AUCTION YOU SHOULD NOT PAY THE AMOUNT GREATER THAN YOU HAVE IN THE PURSE FOR A PARTICULAR PLAYER.

```

Object Explorer   IPL/postgres@DATABASE*  public.ipl_ball/IPL/...  public.ipl_matches...  public.batting_stri...
Query  Query History
121 b.total_wickets
122 from batting_strike_rate as a
123 left join bowling_strikerate as b
124 on a.batsman=b.bowler order by batsman;
125
126 alter table allrounder rename column batsman to player;
127
128 < select * ,balls_bowled*1./nullif(total_wickets,0)as bowling_strike_rate from allrounder
129 where balls_faced > 500 and balls_bowled > 300
130 order by batting_strike_rate desc limit 10;
131
Data Output  Messages  Notifications

```

	player	runs_scored	balls_faced	batting_strike_rate	balls_bowled	total_wickets	bowling_strike_rate
1	AD Russell	1517	832	182.3317307692307692	1147	58	19.7758620689655172
2	SP Narine	892	543	164.2725598526703499	2785	110	25.3181818181818182
3	HH Pandya	1349	847	159.2680047225501771	869	41	21.1951219512195122
4	GJ Maxwell	1505	973	154.6762589928057554	546	16	34.12500000000000000000
5	CH Gayle	4772	3179	150.1100975149418056	554	16	34.62500000000000000000
6	KA Pollard	3023	2017	149.8760535448686168	1325	57	23.2456140350877193
7	YK Pathan	3204	2241	142.9718875502008032	1147	36	31.8611111111111111
8	KH Pandya	1000	702	142.4501424501424501	1245	40	31.12500000000000000000
9	JA Morkel	974	686	141.9825072886297376	1723	78	22.0897435897435897
10	Harbhajan Singh	829	600	138.1666666666666667	3374	138	24.4492753623188406



ADDITIONAL QUESTIONS

GET THE COUNT OF CITIES THAT HAVE HOSTED AN IPL MATCH

The screenshot shows a PostgreSQL query editor window with the following details:

- Database:** IPL/postgres@DATABASE*
- Table:** public.ipl_ball/IPL... (selected tab)
- Query:** A multi-step SQL query to calculate the count of distinct cities that have hosted an IPL match.
- Code (Lines 119-133):**

```
119    FROM batting_strike_rate AS a
120    LEFT JOIN bowling_strikerate AS b
121        ON a.batsman=b.bowler ORDER BY batsman;
122
123    ALTER TABLE allrounder RENAME COLUMN batsman TO player;
124
125    SELECT *, balls_bowled * 1.0 / NULLIF(total_wickets, 0) AS bowling_strike_rate
126    FROM allrounder
127    WHERE balls_faced > 500 AND balls_bowled > 300
128    ORDER BY bowling_strike_rate DESC LIMIT 10;
129
130    SELECT COUNT(DISTINCT city) AS city_count
131    FROM ipl_matches
132    WHERE city IS NOT NULL;
```

- Data Output:** A table showing the result of the final query step.

	city_count	bigint
1	33	

- Messages:** Total rows: 1 of 1 | Query complete 00:00:00.081 | Ln 132, Col 1

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* (BOUNDARY FOR ≥ 4 , DOT FOR 0 AND OTHER FOR ANY OTHER NUMBER)

```

IPL/postgres@DATABASE* X public.ipl_ball/IPL/... X public.ipl_matches... X public.batting_stri...
IPL/postgres@DATABASE
FROM ipl_matches
WHERE city IS NOT NULL;

CREATE TABLE deliveries_v02 AS
SELECT *, 
CASE
    WHEN total_runs >= 4 THEN 'boundary'
    WHEN total_runs = 0 THEN 'dot'
    ELSE 'other'
END AS ball_result
FROM ipl_ball;

SELECT*FROM deliveries_v02;

```

	match_id integer	inning integer	overs integer	ball integer	batsman character varying (100)	non_striker character varying (100)	bowler character varying (100)	batsm intege
1	335982	1	6	5	RT Ponting	BB McCullum	AA Noffke	
2	335982	1	6	6	BB McCullum	RT Ponting	AA Noffke	
3	335982	1	7	1	BB McCullum	RT Ponting	Z Khan	
4	335982	1	7	2	BB McCullum	RT Ponting	Z Khan	
5	335982	1	7	3	RT Ponting	BB McCullum	Z Khan	
6	335982	1	7	4	BB McCullum	RT Ponting	Z Khan	
7	335982	1	7	5	RT Ponting	BB McCullum	Z Khan	
8	335982	1	7	6	BB McCullum	RT Ponting	Z Khan	
9	335982	1	8	1	BB McCullum	RT Ponting	JH Kallis	

Total rows: 1000 of 193468 Query complete 00:00:01.523 Ln 143, Col 28

WRITE A QUERY TO FETCH
THE TOTAL NUMBER OF
BOUNDARIES AND DOT
BALLS FROM THE
DELIVERIES_V02 TABLE.

IPL/postgres@DATABASE* X public.ipl_ball/IPL/... X public.ipl_matches... X public.batting_stri... X

Query History

```
135 SELECT *,
136     CASE
137         WHEN total_runs >= 4 THEN 'boundary'
138         WHEN total_runs = 0 THEN 'dot'
139         ELSE 'other'
140     END AS ball_result
141 FROM ipl_ball;
142
143 SELECT*FROM deliveries_v02;
144
145 SELECT
146     SUM(CASE WHEN ball_result = 'boundary' THEN 1 ELSE 0 END) AS total_boundaries,
147     SUM(CASE WHEN ball_result = 'dot' THEN 1 ELSE 0 END) AS total_dot_balls
148 FROM deliveries_v02;
```

Data Output Messages Notifications

	total_boundaries	total_dot_balls
1	31468	67841

Total rows: 1 of 1 Query complete 00:00:00.503 Ln 145, Col 1

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.

IPL/postgres@DATABASE* X public.ipl_ball/IPL/... X public.ipl_matches... X public.batting_stri... X

Query History

```
143 SELECT*FROM deliveries_v02;
144
145 < SELECT
146     SUM(CASE WHEN ball_result = 'boundary' THEN 1 ELSE 0 END) AS total_boundaries,
147     SUM(CASE WHEN ball_result = 'dot' THEN 1 ELSE 0 END) AS total_dot_balls
148 FROM deliveries_v02;
149
150 < SELECT
151     batting_team,
152     COUNT(*) AS total_boundaries
153 FROM deliveries_v02
154 WHERE ball_result = 'boundary'
155 GROUP BY batting_team
156 ORDER BY total_boundaries DESC;
```

Data Output Messages Notifications

	batting_team character varying (100)	total_boundaries 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

Total rows: 15 of 15 Query complete 00:00:00.477 Ln 150, Col 1

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.

IPL/postgres@DATABASE* X public.ipl_ball/IPL/... X public.ipl_matches... X public.batting_stri... X

Query History

```
151     batting_team,
152     COUNT(*) AS total_boundaries
153   FROM deliveries_v02
154  WHERE ball_result = 'boundary'
155  GROUP BY batting_team
156 ORDER BY total_boundaries DESC;
157
158 <-- SELECT
159     bowling_team,
160     COUNT(*) AS total_dot_balls
161   FROM deliveries_v02
162  WHERE ball_result = 'dot'
163  GROUP BY bowling_team
164 ORDER BY total_dot_balls DESC;
```

Data Output Messages Notifications

	bowling_team character varying (100)	total_dot_balls bigint
1	Mumbai Indians	8714
2	Royal Challengers Bangalore	7955
3	Kolkata Knight Riders	7894
4	Kings XI Punjab	7679
5	Chennai Super Kings	7593
6	Rajasthan Royals	6665
7	Delhi Daredevils	6520
8	Sunrisers Hyderabad	5248
9	Deccan Chargers	3306
10	Pune Warriors	1900

Total rows: 16 of 16 Query complete 00:00:00.269 Ln 158, Col 1

WRITE A QUERY TO FETCH
THE TOTAL NUMBER OF
DISMISSALS BY DISMISSAL
KINDS WHERE DISMISSAL
KIND IS NOT NA

IPL/postgres@DATABASE* X public.ipl_ball/IPL... X public.ipl_matches... X public.batting_stri... X

Execute script F5

```
159     bowling_team,
160     COUNT(*) AS total_dot_balls
161   FROM deliveries_v02
162   WHERE ball_result = 'dot'
163   GROUP BY bowling_team
164   ORDER BY total_dot_balls DESC;
165
166   SELECT
167     dismissal_kind,
168     COUNT(*) AS total_dismissals
169   FROM IPL_BALL
170   WHERE dismissal_kind IS NOT NULL AND dismissal_kind != 'NA'
171   GROUP BY dismissal_kind
172   ORDER BY total_dismissals DESC;
```

Data Output Messages Notifications

	dismissal_kind character varying (100)	total_dismissals bigint
1	caught	5743
2	bowled	1700
3	run out	893
4	lbw	571
5	stumped	294
6	caught and bowled	269
7	hit wicket	12
8	retired hurt	11
9	obstructing the field	2

Total rows: 9 of 9 Query complete 00:00:00.227 Ln 173, Col 1

WRITE A QUERY TO GET THE
TOP 5 BOWLERS WHO
CONCEDED MAXIMUM
EXTRA RUNS FROM THE
DELIVERIES TABLE

IPL/postgres@DATABASE* X public.ipl_ball/IPL... X public.ipl_matches... X public.batting_stri... X

Query History

```
168 COUNT(*) AS total_dismissals
169 FROM IPL_BALL
170 WHERE dismissal_kind IS NOT NULL AND dismissal_kind != 'NA'
171 GROUP BY dismissal_kind
172 ORDER BY total_dismissals DESC;
173
174 SELECT
175     bowler,
176     SUM(extra_runs) AS total_extra_runs
177 FROM IPL_BALL
178 GROUP BY bowler
179 ORDER BY total_extra_runs DESC
180 LIMIT 5;
181
182
```

Data Output Messages Notifications

	bowler character varying (100)	total_extra_runs bigint
1	SL Malinga	293
2	P Kumar	236
3	UT Yadav	226
4	DJ Bravo	210
5	B Kumar	201

Total rows: 5 of 5 Query complete 00:00:00.184 Ln 182, Col 1

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*

IPL/postgres@DATABASE* X public.ipl_ball/IPL... X public.ipl_matches... X public.batting_stri... X

Query History

```
176     SUM(extra_runs) AS total_extra_runs
177     FROM IPL_BALL
178     GROUP BY bowler
179     ORDER BY total_extra_runs DESC
180     LIMIT 5;
181
182 ✓ CREATE TABLE deliveries_v03 AS
183     SELECT
184         d.*,
185         m.venue,
186         m.date AS match_date
187     FROM deliveries_v02 d
188     JOIN IPL_matches m ON d.match_id = m.match_id;
189
190     SELECT*FROM deliveries_v03;
```

Data Output Messages Notifications

	match_id integer	inning integer	overs integer	ball integer	batsman character varying (100)	non_striker character varying (100)	bowler character varying (100)	bat... intege
1	335982	1	6	5	RT Ponting	BB McCullum	AA Noffke	
2	335982	1	6	6	BB McCullum	RT Ponting	AA Noffke	
3	335982	1	7	1	BB McCullum	RT Ponting	Z Khan	
4	335982	1	7	2	BB McCullum	RT Ponting	Z Khan	
5	335982	1	7	3	RT Ponting	BB McCullum	Z Khan	
6	335982	1	7	4	BB McCullum	RT Ponting	Z Khan	
7	335982	1	7	5	RT Ponting	BB McCullum	Z Khan	
8	335982	1	7	6	BB McCullum	RT Ponting	Z Khan	
9	335982	1	8	1	BB McCullum	RT Ponting	JH Kallis	

Total rows: 1000 of 193468 Query complete 00:00:01.866 Ln 182, Col 1

WRITE A QUERY TO FETCH THE TOTAL RUNS SCORED FOR EACH VENUE AND ORDER IT IN THE DESCENDING ORDER OF TOTAL RUNS SCORED.

IPL/postgres@DATABASE* X public.ipl_ball/IPL/... X public.ipl_matches... X public.batting_stri... X

Query History

```
184     a.*,
185     m.venue,
186     m.date AS match_date
187   FROM deliveries_v02 d
188   JOIN IPL_matches m ON d.match_id = m.match_id;
189
190   SELECT*FROM deliveries_v03;
191
192 <--> SELECT
193     venue,
194     SUM(total_runs) AS total_runs
195   FROM deliveries_v03
196   GROUP BY venue
197   ORDER BY total_runs DESC;
```

Data Output Messages Notifications

	venue character varying (100)	total_runs bigint
1	Eden Gardens	23658
2	Wankhede Stadium	23390
3	Feroz Shah Kotla	22947
4	M Chinnaswamy Stadium	20237
5	Rajiv Gandhi International Stadium, Uppal	19484
6	MA Chidambaram Stadium, Chepauk	17821
7	Sawai Mansingh Stadium	14264
8	Punjab Cricket Association Stadium, Mohali	10987
9	Dubai International Cricket Stadium	10402
10	Sheikh Zayed Stadium	8830

Total rows: 36 of 36 Query complete 00:00:00.555 Ln 191, Col 1

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.

```
IPL/postgres@DATABASE* X public.ipl_ball/IPL/... X public.ipl_matches... X public.batting_stri... X ...  
IPL/postgres@DATABASE  
No limit  
Query History  
192 SELECT  
193     venue,  
194     SUM(total_runs) AS total_runs  
195 FROM deliveries_v03  
196 GROUP BY venue  
197 ORDER BY total_runs DESC;  
198  
199 SELECT  
200     EXTRACT(YEAR FROM match_date) AS year,  
201     SUM(total_runs) AS total_runs  
202 FROM deliveries_v03  
203 WHERE venue = 'Eden Gardens'  
204 GROUP BY year  
205 ORDER BY total_runs DESC;  
206  
Data Output Messages Notifications  
year numeric total_runs bigint  
1 2018 2885  
2 2019 2651  
3 2015 2386  
4 2013 2304  
5 2017 2194  
6 2010 2167  
7 2016 2073  
8 2012 2012  
9 2011 1854  
10 2008 1843  
Total rows: 11 of 11 Query complete 00:00:00.512  
Ln 199, Col 1
```

	year	total_runs
	numeric	bigint
1	2018	2885
2	2019	2651
3	2015	2386
4	2013	2304
5	2017	2194
6	2010	2167
7	2016	2073
8	2012	2012
9	2011	1854
10	2008	1843

THANK YOU

