Mini-Project [DBMS] (GROUP-5)

Great Learning
POWER AHEAD

TITLE :Pie-in-the-Sky(IPL Match Bidding App)

Submitted by : Manjunath I Kunte

Munir Sheth

Muzammil Firdous M A

Nandish S Pattanashetti

Nandakumar

Problem Definition

- Dataset:

Pie-in-the-Sky is a mobile app that is used for bidding for IPL matches legally. Any registered user can bid for any of the IPL matches listed in it. New users or bidders need to register themselves into the app by providing their mobile phone number, email id, and password. Admin will maintain the match roster and keep updating other details in the system.

The app shows the match details which include the playing team, the venue of the match, and the current standing of the teams on the points table. It will display the winner at the end of the match and update the team standings in the tournament and bidder points table. System will send updates to the bidders whenever required. It will also generate the bidders' leaderboard.

- Problem Statement :

The problem statement is to use the SQL queries to find the various insight from the above-given data. And also write your insights based on the results that you will get from the queries which you will be using.

Business Importance of Problem

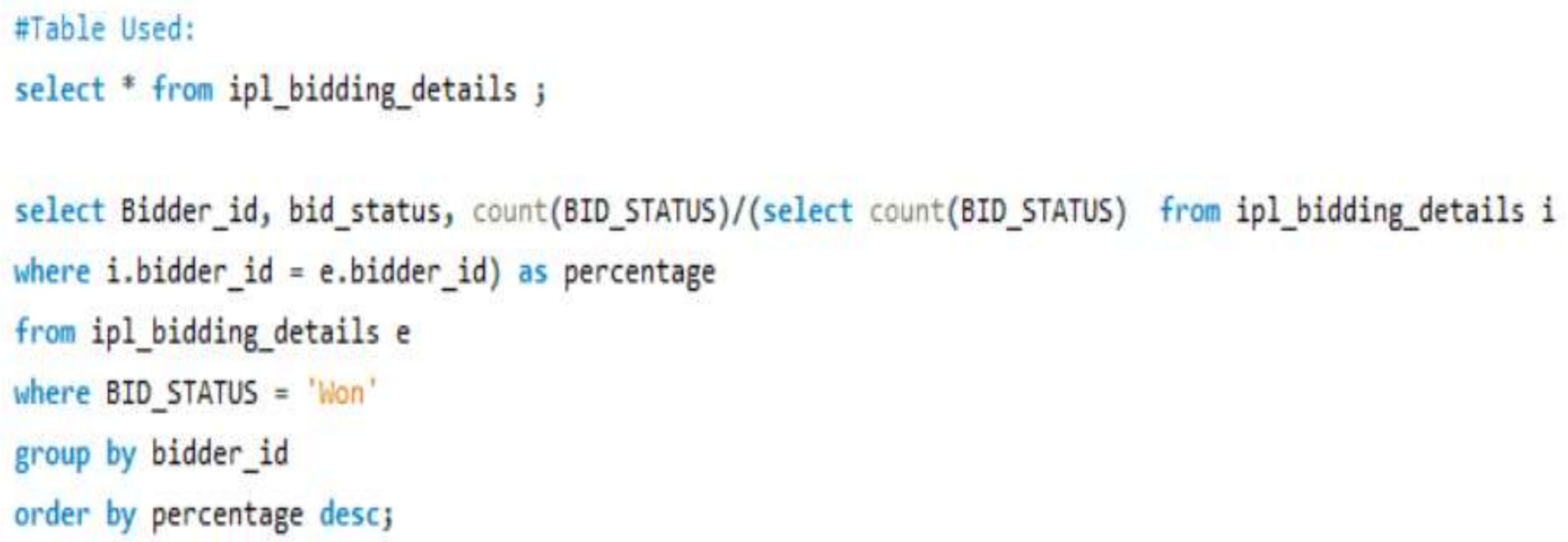
The provided information about the "Pie-in-the-Sky" IPL bidding app and its associated database is important for businesses as it offers a platform for legal bidding on IPL matches. This app can provide a competitive and engaging experience for users by allowing them to predict match winners, earn points, and compete on leaderboards. Additionally, the app's database stores valuable data on user preferences, team standings, match results, and more, which can be analyzed to gain insights into user behavior, popular teams, and match outcomes. This information can help businesses tailor marketing strategies, optimize user engagement, and make informed decisions related to team sponsorships, promotions, and user retention strategies.

Q1. Show the percentage of wins of each bidder in the order of highest to lowest percentage.

The question is asking you to display the percentage of wins for each bidder in a certain context from Descending to Ascending order

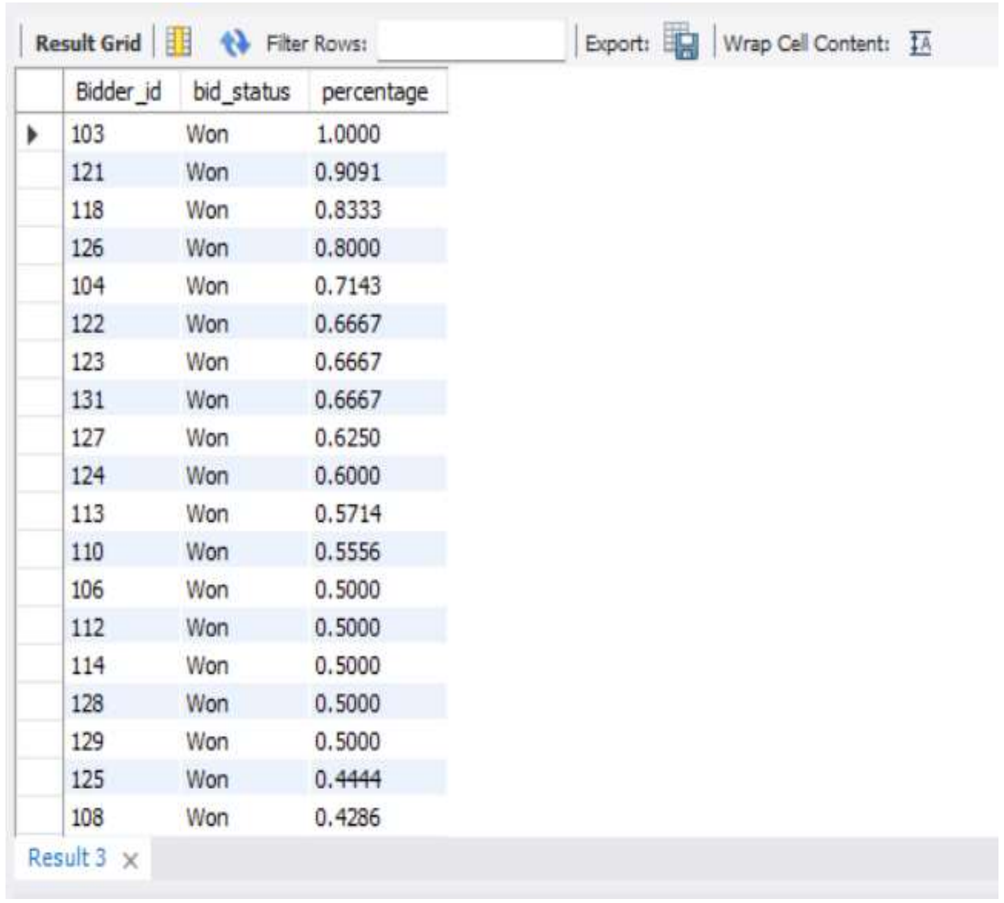
1. Gather Data
2. Calculate Percentages
3. Sort the Bidders
4. Display the Results

Code:

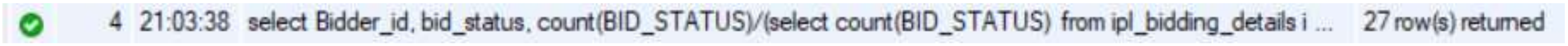


```
#Table Used:
select * from ipl_bidding_details ;

select Bidder_id, bid_status, count(BID_STATUS)/(select count(BID_STATUS)  from ipl_bidding_details i
where i.bidder_id = e.bidder_id) as percentage
from ipl_bidding_details e
where BID_STATUS = 'Won'
group by bidder_id
order by percentage desc;
```

Output:



Bidder_id	bid_status	percentage
103	Won	1.0000
121	Won	0.9091
118	Won	0.8333
126	Won	0.8000
104	Won	0.7143
122	Won	0.6667
123	Won	0.6667
131	Won	0.6667
127	Won	0.6250
124	Won	0.6000
113	Won	0.5714
110	Won	0.5556
106	Won	0.5000
112	Won	0.5000
114	Won	0.5000
128	Won	0.5000
129	Won	0.5000
125	Won	0.4444
108	Won	0.4286



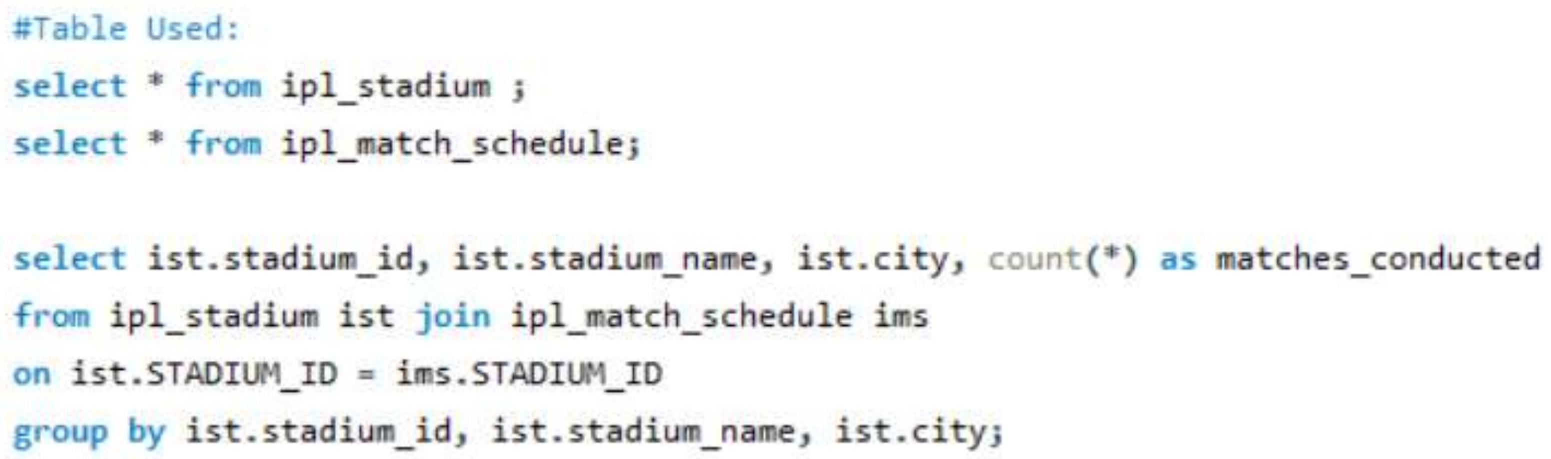
```
4  21:03:38  select Bidder_id, bid_status, count(BID_STATUS)/(select count(BID_STATUS) from ipl_bidding_details i ...  27 row(s) returned
```

Q2. Display the number of matches conducted at each stadium with the stadium name and city.

The question is asking you to create a display that shows the number of matches conducted at each stadium, along with the corresponding stadium name and city.

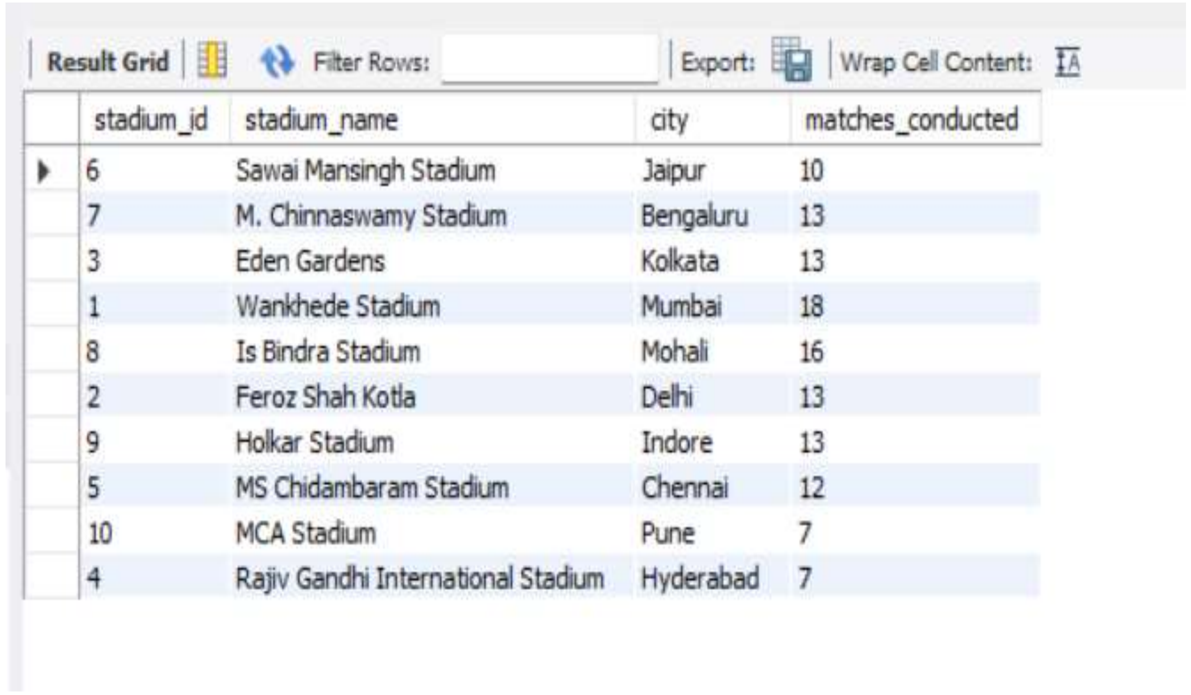
1. Data Source
2. Data Preparation
3. Data Analysis
4. Output

Code:



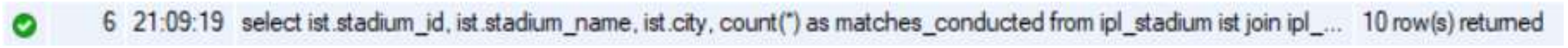
```
#Table Used:
select * from ipl_stadium ;
select * from ipl_match_schedule;

select ist.stadium_id, ist.stadium_name, ist.city, count(*) as matches_conducted
from ipl_stadium ist join ipl_match_schedule ims
on ist.STADIUM_ID = ims.STADIUM_ID
group by ist.stadium_id, ist.stadium_name, ist.city;
```

Output:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

stadium_id	stadium_name	city	matches_conducted
6	Sawai Mansingh Stadium	Jaipur	10
7	M. Chinnaswamy Stadium	Bengaluru	13
3	Eden Gardens	Kolkata	13
1	Wankhede Stadium	Mumbai	18
8	Is Bindra Stadium	Mohali	16
2	Feroz Shah Kotla	Delhi	13
9	Holkar Stadium	Indore	13
5	MS Chidambaram Stadium	Chennai	12
10	MCA Stadium	Pune	7
4	Rajiv Gandhi International Stadium	Hyderabad	7



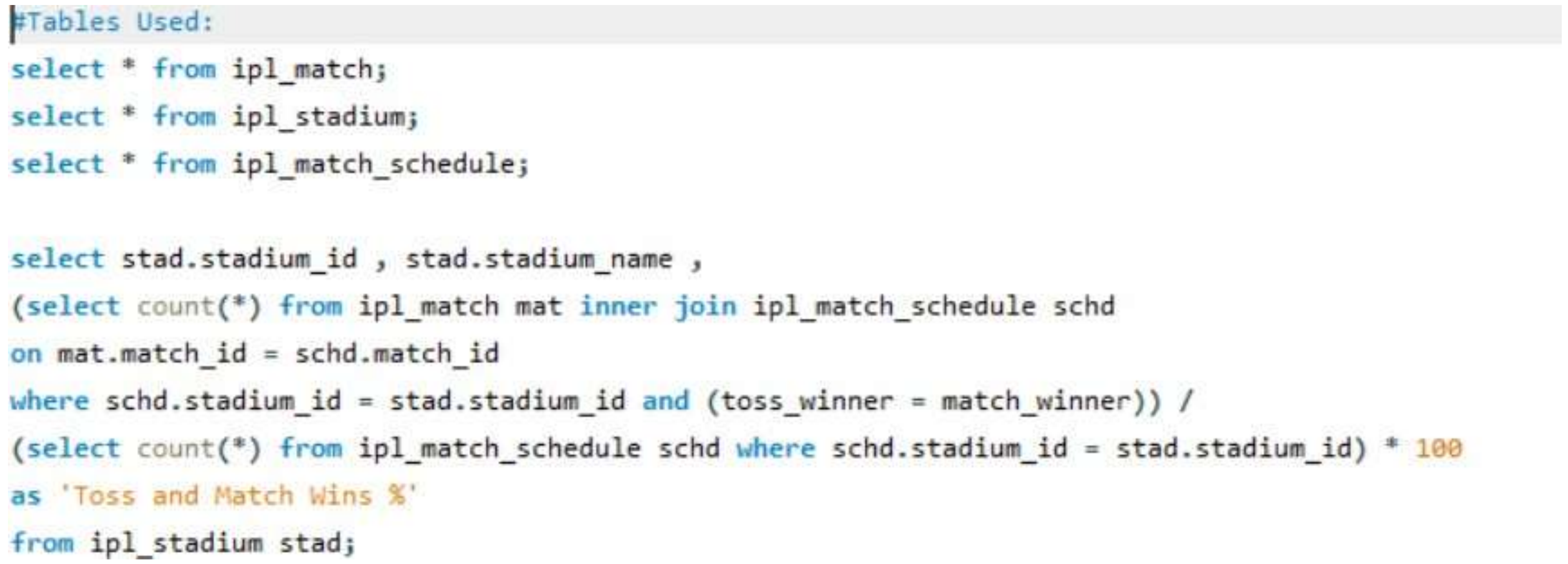
```
6  21:09:19  select ist.stadium_id, ist.stadium_name, ist.city, count(*) as matches_conducted from ipl_stadium ist join ipl_...  10 row(s) returned
```

Q3. In a given stadium, what is the percentage of wins by a team which has won the toss?

The question is asking for the percentage of wins by a specific team in a given stadium, but with a condition: the team should have won the toss in those matches they went on to win

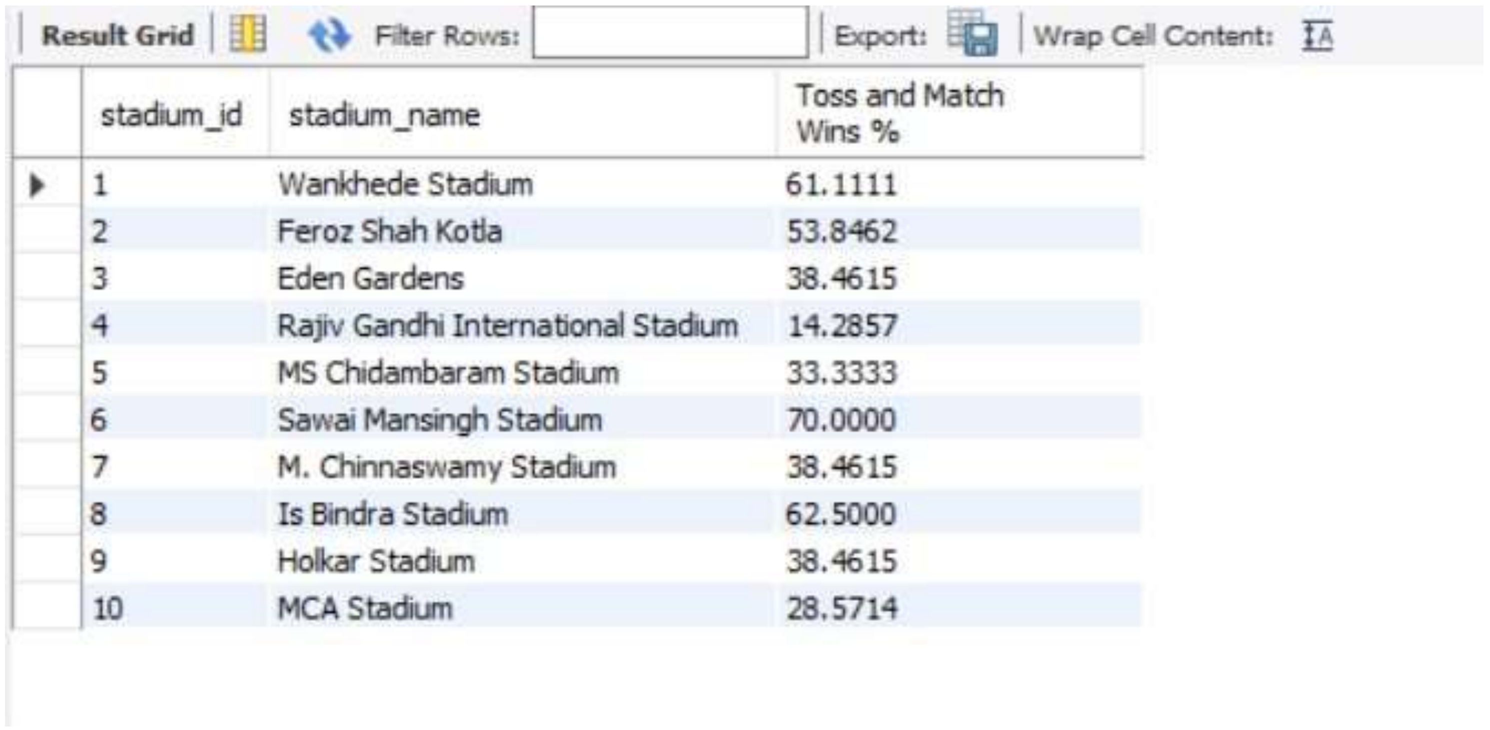
1. Stadium
2. Team
3. Toss
4. Wins

Code:

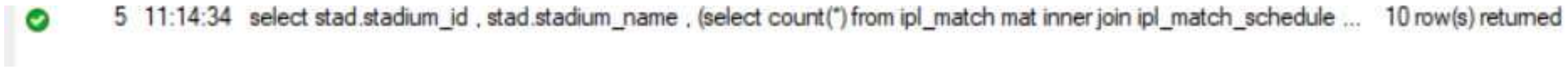


```
#Tables Used:
select * from ipl_match;
select * from ipl_stadium;
select * from ipl_match_schedule;

select stad.stadium_id , stad.stadium_name ,
(select count(*) from ipl_match mat inner join ipl_match_schedule schd
on mat.match_id = schd.match_id
where schd.stadium_id = stad.stadium_id and (toss_winner = match_winner)) /
(select count(*) from ipl_match_schedule schd where schd.stadium_id = stad.stadium_id) * 100
as 'Toss and Match Wins %'
from ipl_stadium stad;
```

Output:



	stadium_id	stadium_name	Toss and Match Wins %
▶	1	Wankhede Stadium	61.1111
	2	Feroz Shah Kotla	53.8462
	3	Eden Gardens	38.4615
	4	Rajiv Gandhi International Stadium	14.2857
	5	MS Chidambaram Stadium	33.3333
	6	Sawai Mansingh Stadium	70.0000
	7	M. Chinnaswamy Stadium	38.4615
	8	Is Bindra Stadium	62.5000
	9	Holkar Stadium	38.4615
	10	MCA Stadium	28.5714



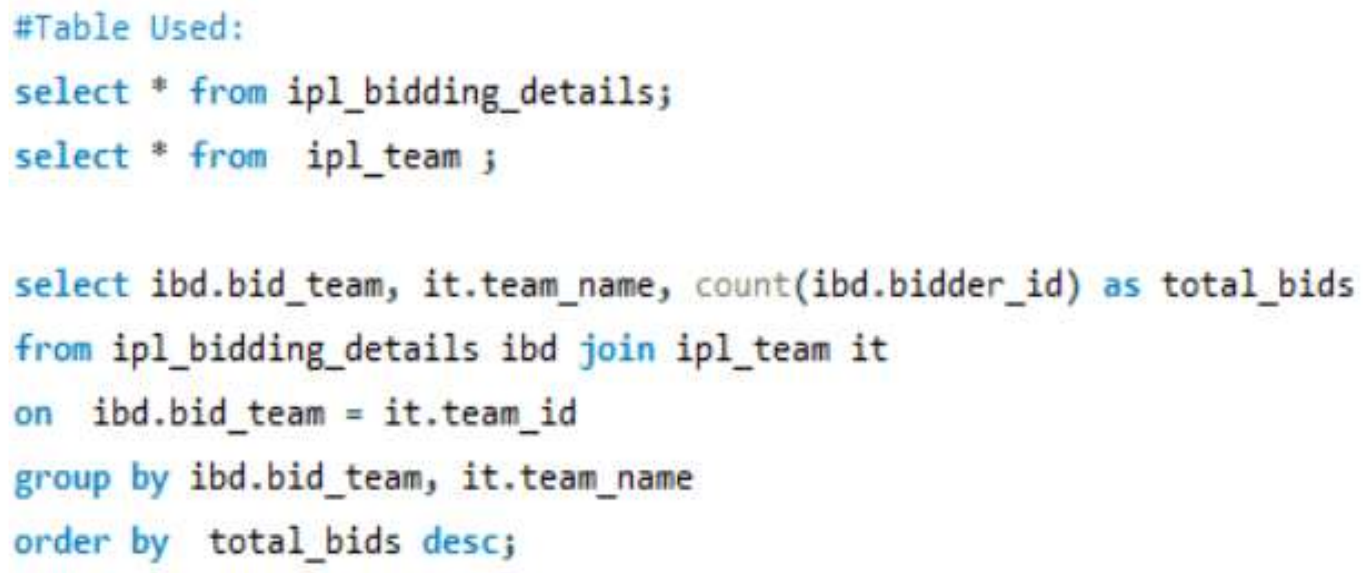
```
5  11:14:34  select stad.stadium_id , stad.stadium_name , (select count(*) from ipl_match mat inner join ipl_match_schedule ...   10 row(s) returned
```

Q 4. Show the total bids along with the bid team and team name.

We are being asked to present information about the total bids, the team responsible for each bid, and the names of those teams

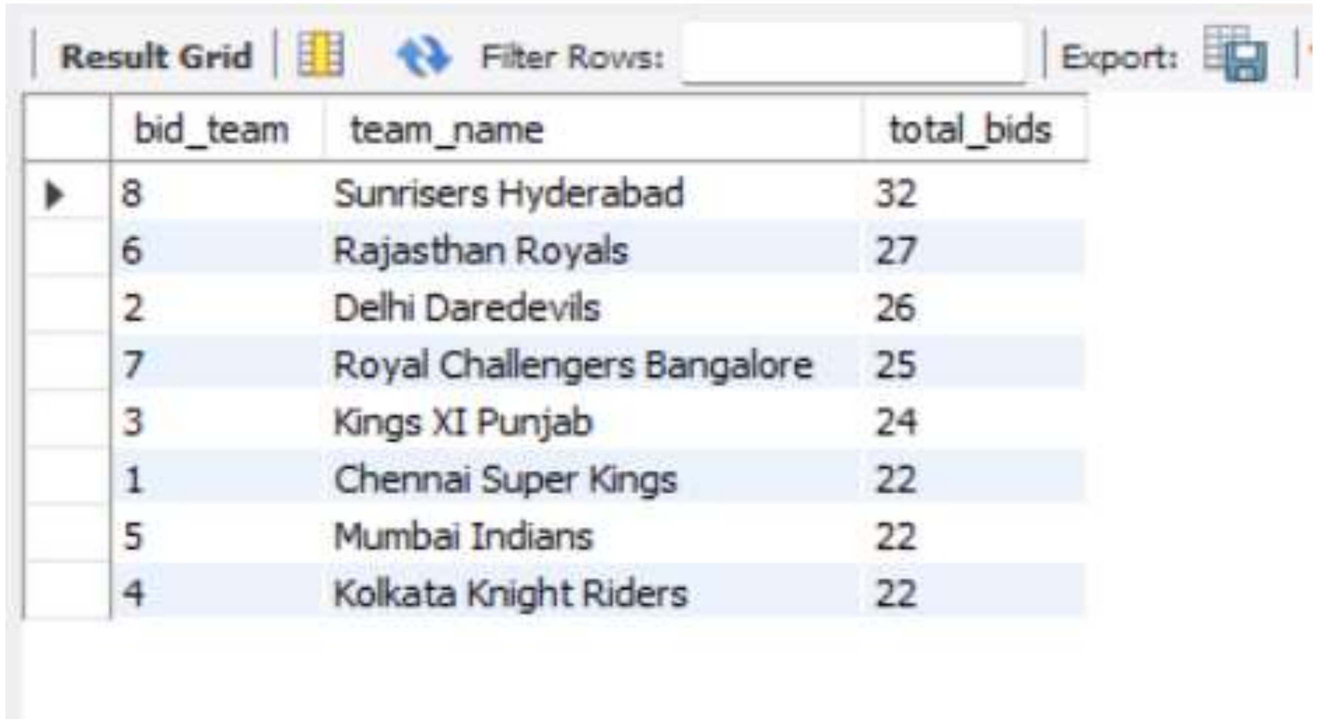
1. Total Bids
2. Bid Team
3. Team Name
4. Output

Code:



```
#Table Used:
select * from ipl_bidding_details;
select * from  ipl_team ;

select ibd.bid_team, it.team_name, count(ibd.bidder_id) as total_bids
from ipl_bidding_details ibd join ipl_team it
on  ibd.bid_team = it.team_id
group by ibd.bid_team, it.team_name
order by  total_bids desc;
```

Output:

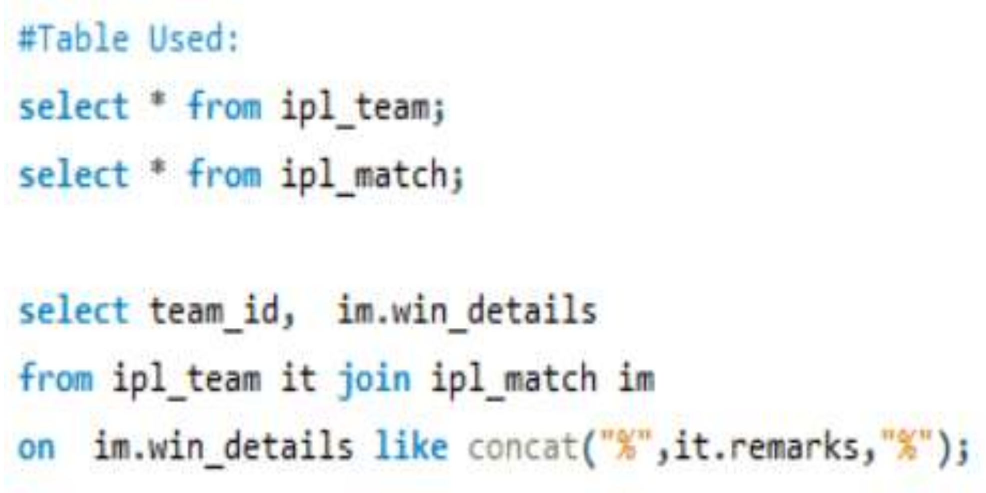

Result Grid | Filter Rows: | Export:

bid_team	team_name	total_bids
8	Sunrisers Hyderabad	32
6	Rajasthan Royals	27
2	Delhi Daredevils	26
7	Royal Challengers Bangalore	25
3	Kings XI Punjab	24
1	Chennai Super Kings	22
5	Mumbai Indians	22
4	Kolkata Knight Riders	22

Q 5. Show the team id who won the match as per the win details.

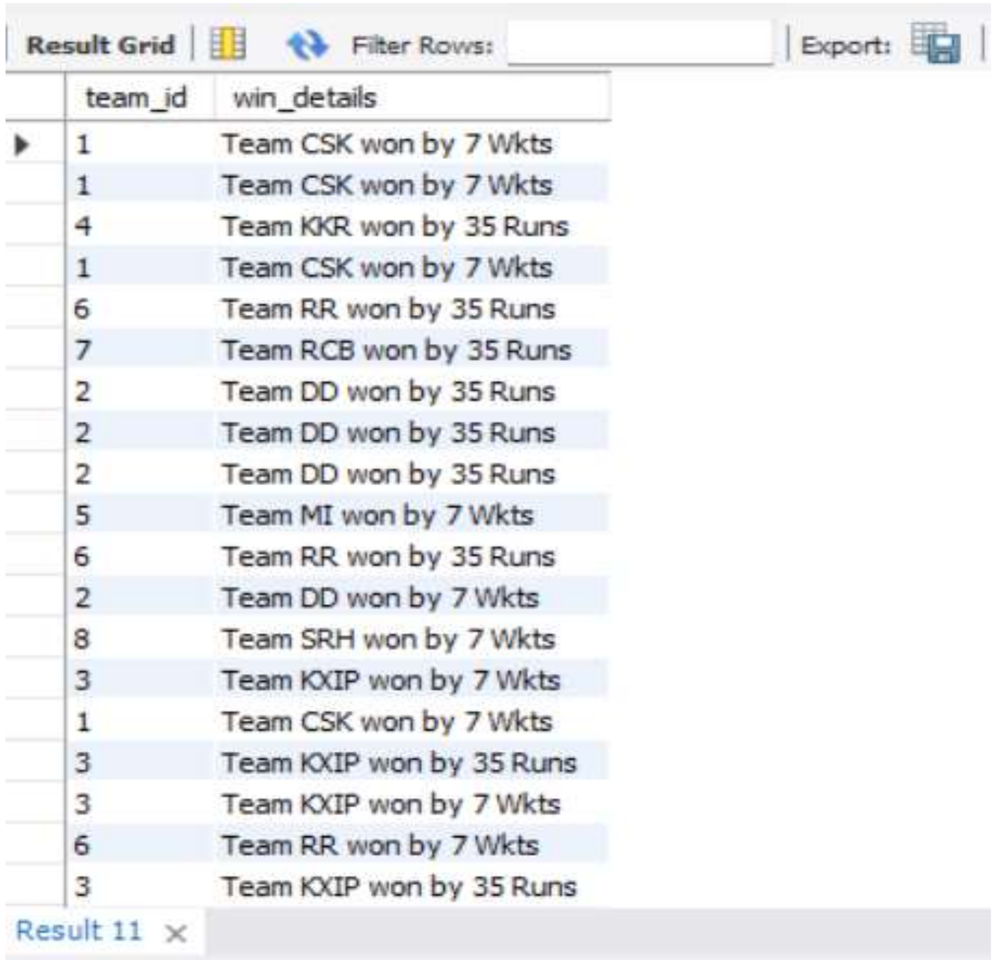
The question is asking you to display or provide information about the team ID (identification) of the team that won a match based on the "win details.”

Code:

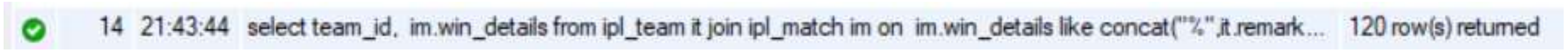


```
#Table Used:
select * from ipl_team;
select * from ipl_match;

select team_id,  im.win_details
from ipl_team it join ipl_match im
on  im.win_details like concat("%",it.remarks,"%");
```

Output:



team_id	win_details
1	Team CSK won by 7 Wkts
1	Team CSK won by 7 Wkts
4	Team KKR won by 35 Runs
1	Team CSK won by 7 Wkts
6	Team RR won by 35 Runs
7	Team RCB won by 35 Runs
2	Team DD won by 35 Runs
2	Team DD won by 35 Runs
2	Team DD won by 35 Runs
5	Team MI won by 7 Wkts
6	Team RR won by 35 Runs
2	Team DD won by 7 Wkts
8	Team SRH won by 7 Wkts
3	Team KXIP won by 7 Wkts
1	Team CSK won by 7 Wkts
3	Team KXIP won by 35 Runs
3	Team KXIP won by 7 Wkts
6	Team RR won by 7 Wkts
3	Team KXIP won by 35 Runs



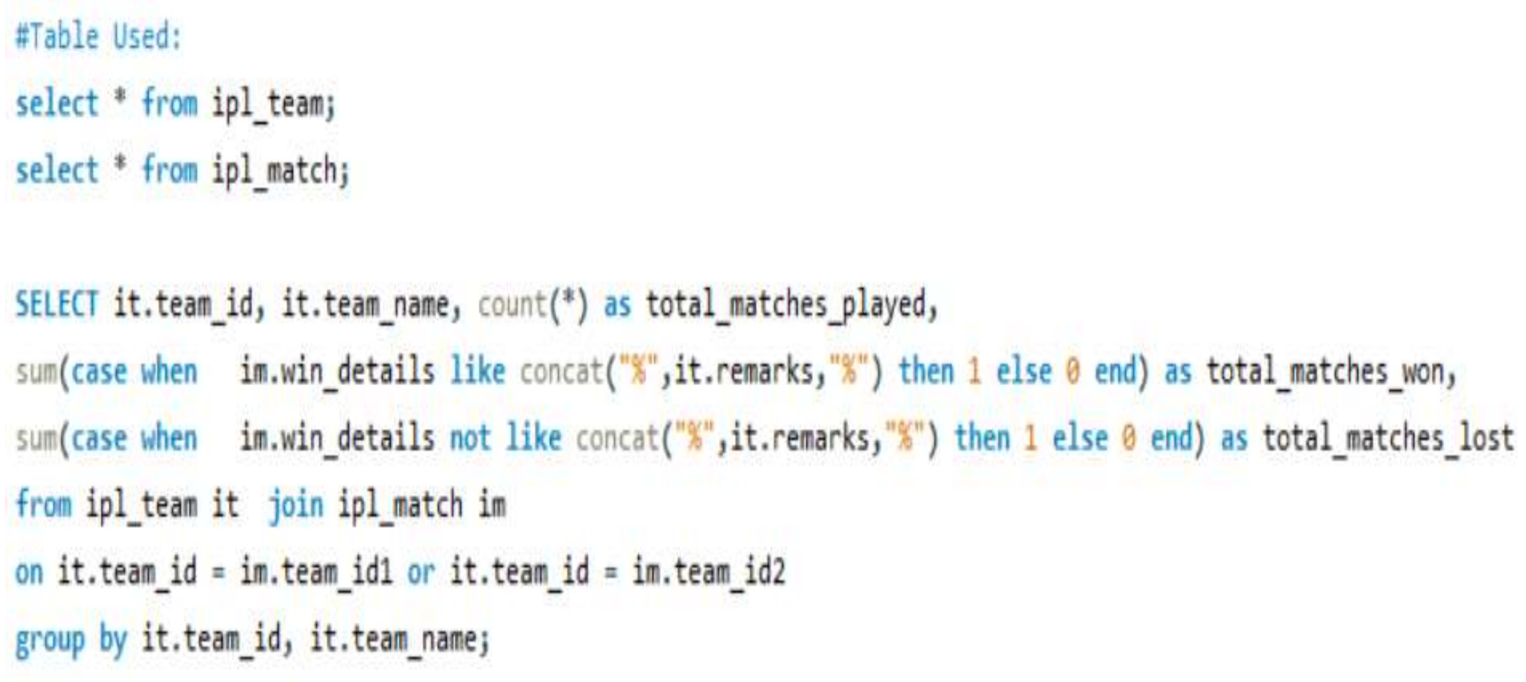
```
14  21:43:44  select team_id,  im.win_details from ipl_team it join ipl_match im on  im.win_details like concat("%",it.remark...  120 row(s) returned
```

Q6. Display total matches played, total matches won and total matches lost by the team along with its team name.

This question is asking you to present a summary of a team's performance in a set of matches. The summary should include the following information for the specified team:

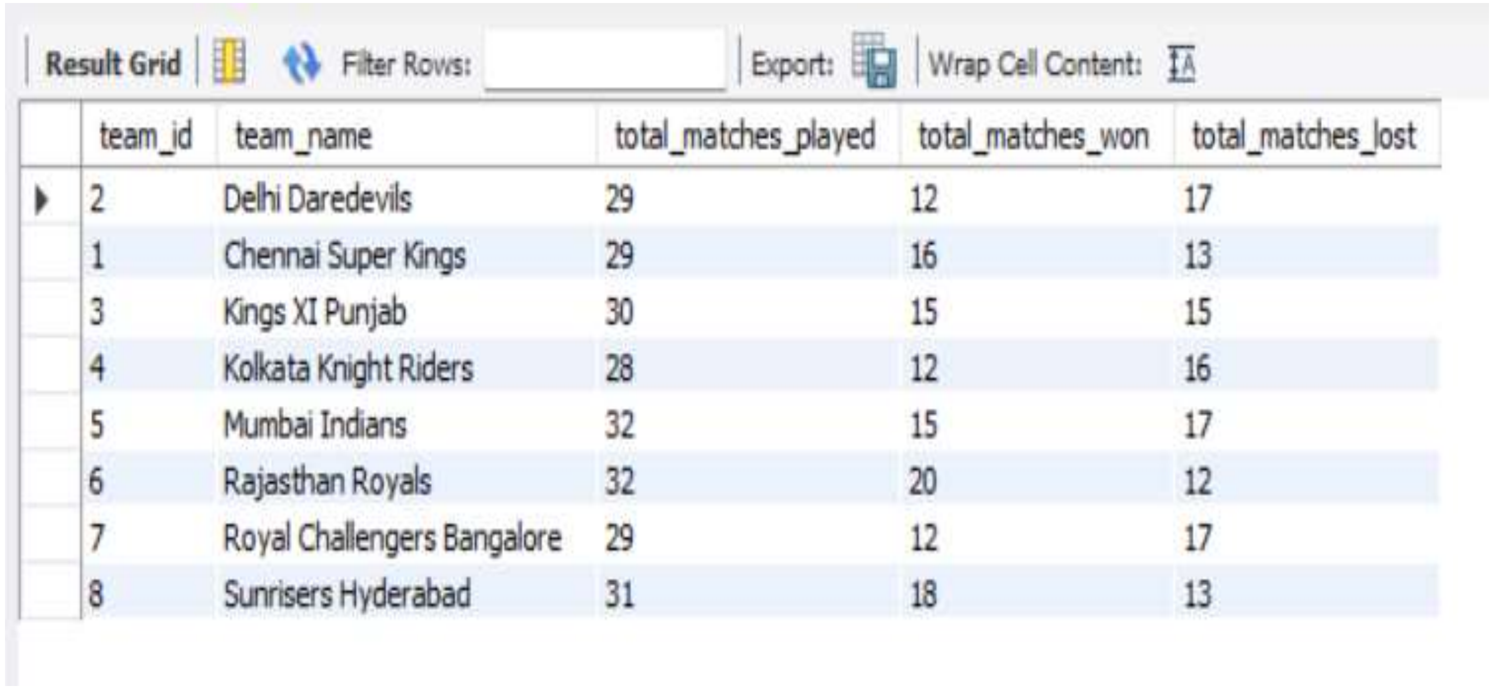
1. Total matches played.
2. Total matches won.
3. Total matches lost.
4. The team's name.

Code:



```
#Table Used:
select * from ipl_team;
select * from ipl_match;

SELECT it.team_id, it.team_name, count(*) as total_matches_played,
sum(case when   im.win_details like concat("%",it.remarks,"%") then 1 else 0 end) as total_matches_won,
sum(case when   im.win_details not like concat("%",it.remarks,"%") then 1 else 0 end) as total_matches_lost
from ipl_team it  join ipl_match im
on it.team_id = im.team_id1 or it.team_id = im.team_id2
group by it.team_id, it.team_name;
```

Output:

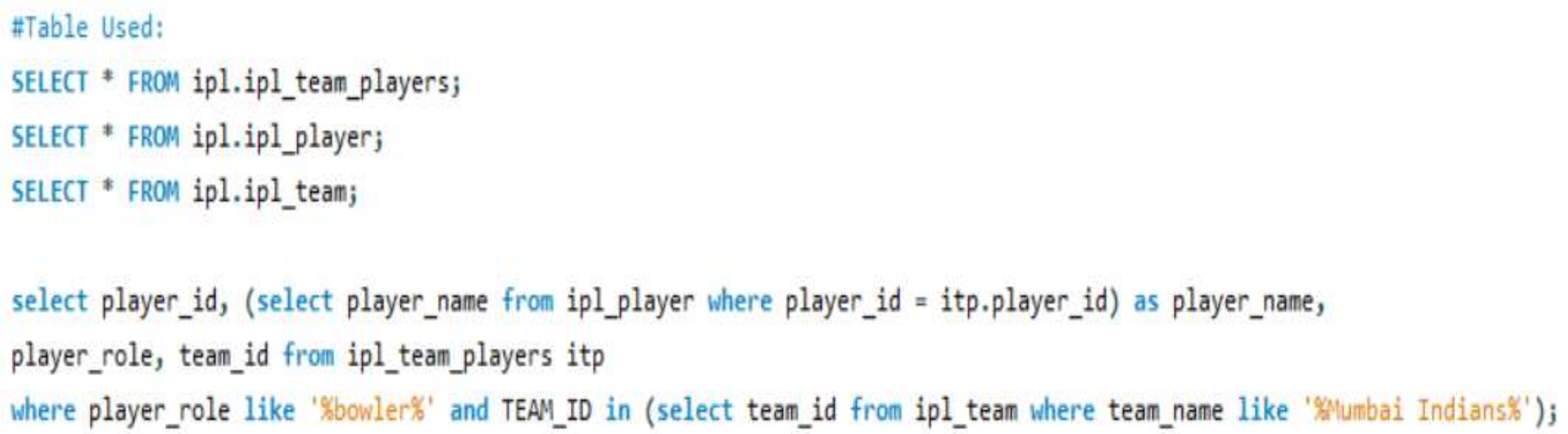

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

team_id	team_name	total_matches_played	total_matches_won	total_matches_lost
2	Delhi Daredevils	29	12	17
1	Chennai Super Kings	29	16	13
3	Kings XI Punjab	30	15	15
4	Kolkata Knight Riders	28	12	16
5	Mumbai Indians	32	15	17
6	Rajasthan Royals	32	20	12
7	Royal Challengers Bangalore	29	12	17
8	Sunrisers Hyderabad	31	18	13

Q 7. Display the bowlers for the Mumbai Indians team.

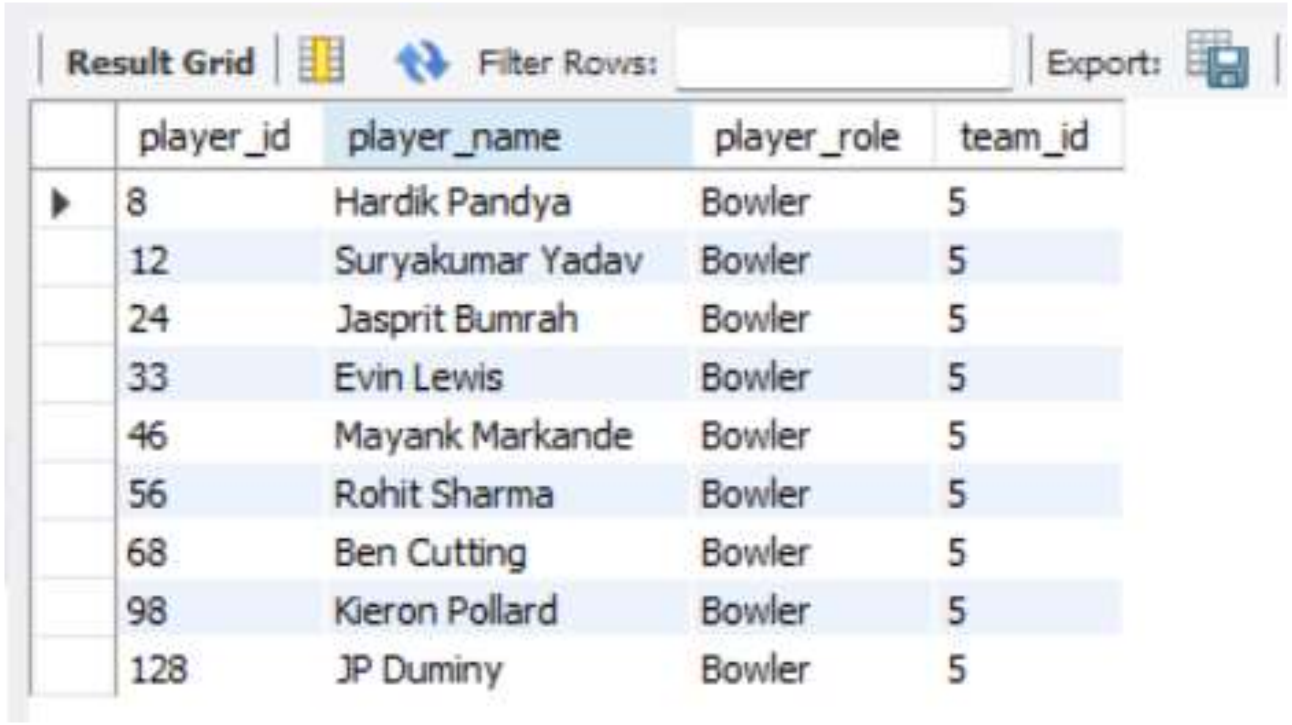
This question is asking to present a list of bowlers who belong to the "Mumbai Indians" cricket team.

Code:



```
#Table Used:
SELECT * FROM ipl.ipl_team_players;
SELECT * FROM ipl.ipl_player;
SELECT * FROM ipl.ipl_team;

select player_id, (select player_name from ipl_player where player_id = itp.player_id) as player_name,
player_role, team_id from ipl_team_players itp
where player_role like '%bowler%' and TEAM_ID in (select team_id from ipl_team where team_name like '%Mumbai Indians%');
```

Output:



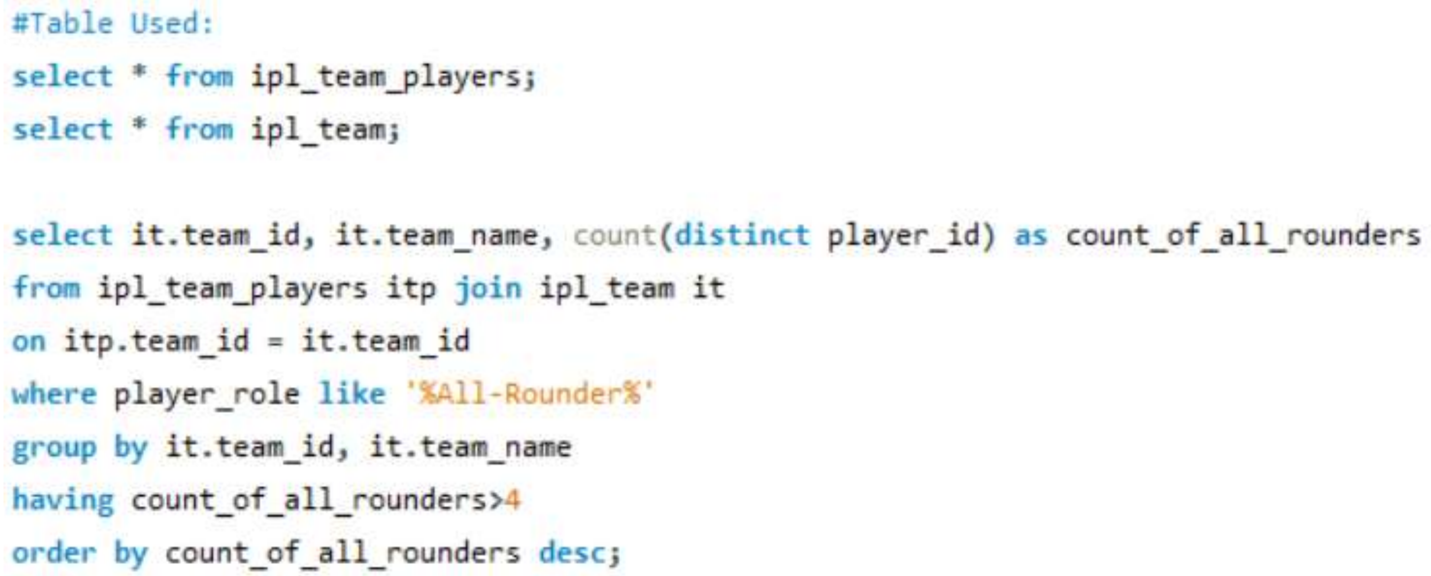
	player_id	player_name	player_role	team_id
▶	8	Hardik Pandya	Bowler	5
	12	Suryakumar Yadav	Bowler	5
	24	Jasprit Bumrah	Bowler	5
	33	Evin Lewis	Bowler	5
	46	Mayank Markande	Bowler	5
	56	Rohit Sharma	Bowler	5
	68	Ben Cutting	Bowler	5
	98	Kieron Pollard	Bowler	5
	128	JP Duminy	Bowler	5

Q 8. How many all-rounders are there in each team, Display the teams with more than 4 all-rounders in descending order.

The question is asking for information about the number of all-rounders in each team and then requesting the display of teams that have more than 4 all-rounders, listed in descending order based on the number of all-rounders.

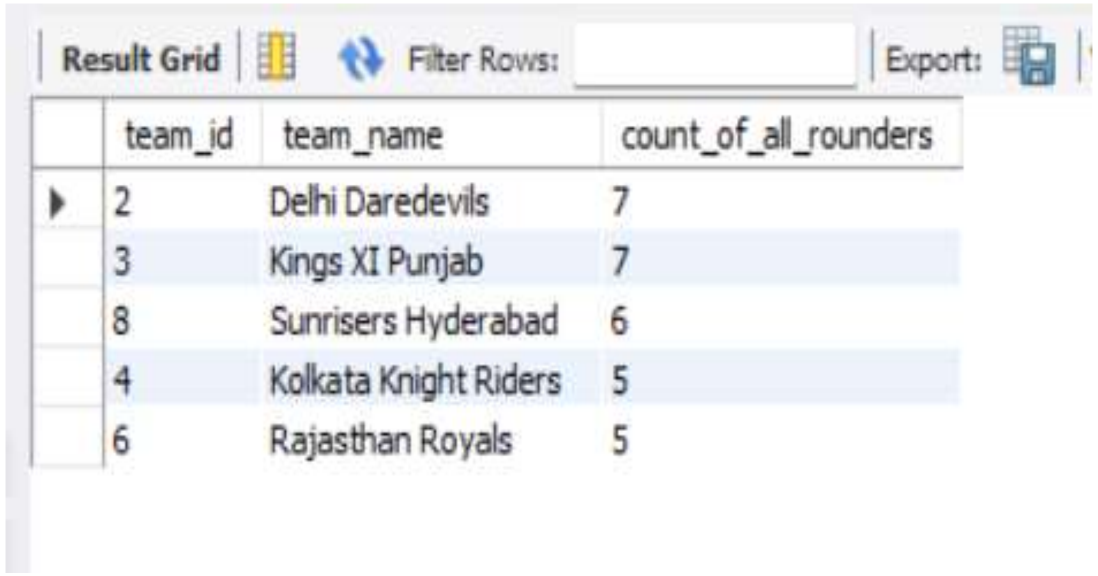
1. Count All-Rounders
2. Identify Teams
3. Filter Teams
4. Descending Order

Code:



```
#Table Used:
select * from ipl_team_players;
select * from ipl_team;

select it.team_id, it.team_name, count(distinct player_id) as count_of_all_rounders
from ipl_team_players itp join ipl_team it
on itp.team_id = it.team_id
where player_role like '%All-Rounder%'
group by it.team_id, it.team_name
having count_of_all_rounders>4
order by count_of_all_rounders desc;
```

Output:



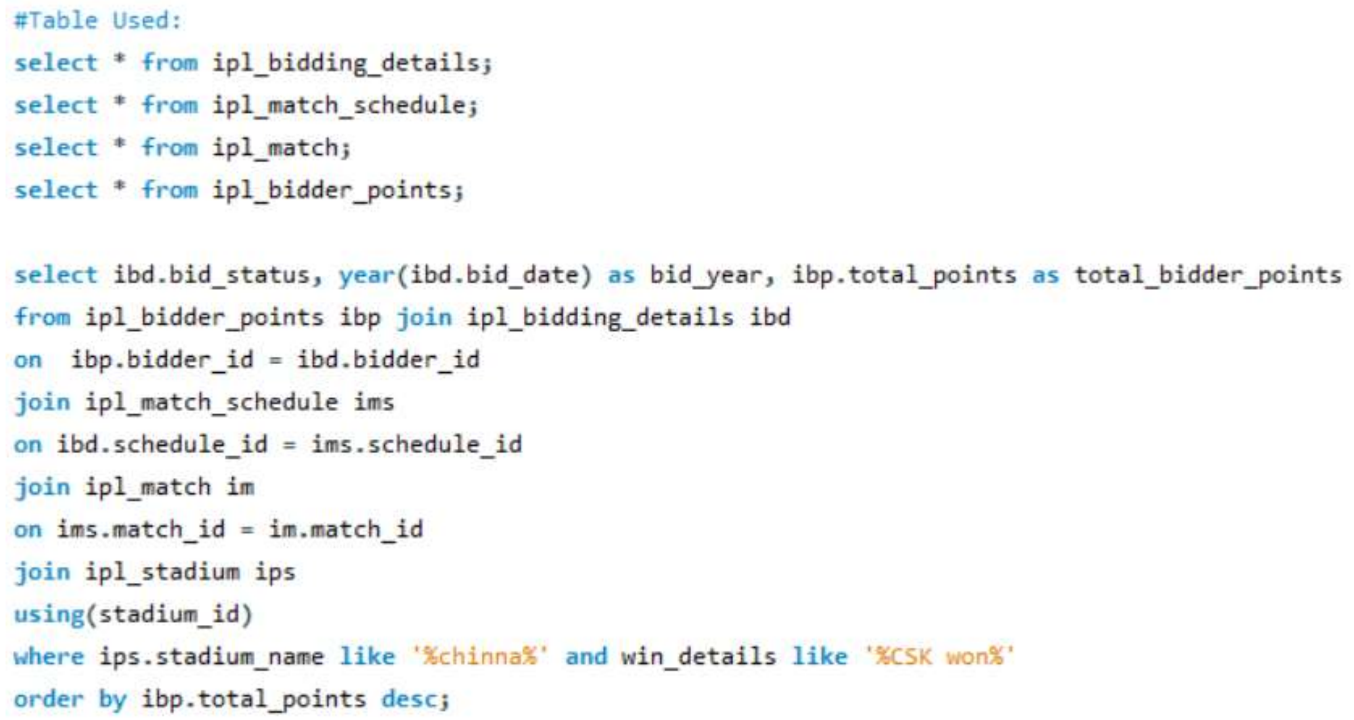
team_id	team_name	count_of_all_rounders
2	Delhi Daredevils	7
3	Kings XI Punjab	7
8	Sunrisers Hyderabad	6
4	Kolkata Knight Riders	5
6	Rajasthan Royals	5

Q 9.Write a query to get the total bidders points for each bidding status of those bidders who bid on CSK when it won the match in M. Chinnaswamy Stadium bidding year-wise.Note the total bidders' points in descending order and the year is bidding year.Display columns: bidding status, bid date as year, total bidder's points

This question is asking you to write a database query that retrieves information about bidders who placed bids on the CSK cricket team when they won matches at the M. Chinnaswamy Stadium. The query should group the data by bidding status and bidding year, and then display the total bidder's points for each combination of bidding status and bidding year. The results should be sorted in descending order based on the total bidder's points. The requested columns in the output are: bidding status, bid date (extracted as year), and total bidder's points.

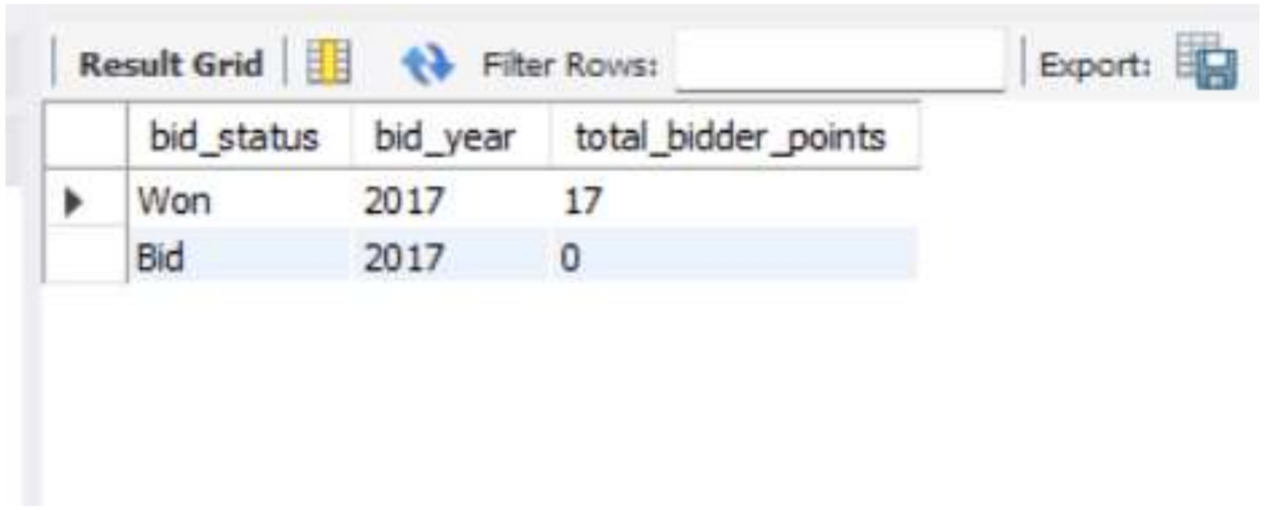
1. Filter Bids
2. Grouping
3. Calculate Total Points
4. Ordering
5. Columns in Output

Code:



```
#Table Used:
select * from ipl_bidding_details;
select * from ipl_match_schedule;
select * from ipl_match;
select * from ipl_bidder_points;

select ibd.bid_status, year(ibd.bid_date) as bid_year, ibp.total_points as total_bidder_points
from ipl_bidder_points ibp join ipl_bidding_details ibd
on  ibp.bidder_id = ibd.bidder_id
join ipl_match_schedule ims
on ibd.schedule_id = ims.schedule_id
join ipl_match im
on ims.match_id = im.match_id
join ipl_stadium ips
using(stadium_id)
where ips.stadium_name like '%chinna%' and win_details like '%CSK won%'
order by ibp.total_points desc;
```

Output:



Result Grid | Filter Rows: | Export:

bid_status	bid_year	total_bidder_points
Won	2017	17
Bid	2017	0

Q10. Extract the Bowlers and All Rounders those are in the 5 highest number of wickets.

-- Note

1.use the performance_dtls column from ipl_player to get the total number of wickets

2.Do not use the limit method because it might not give appropriate results when players have the same number of wickets

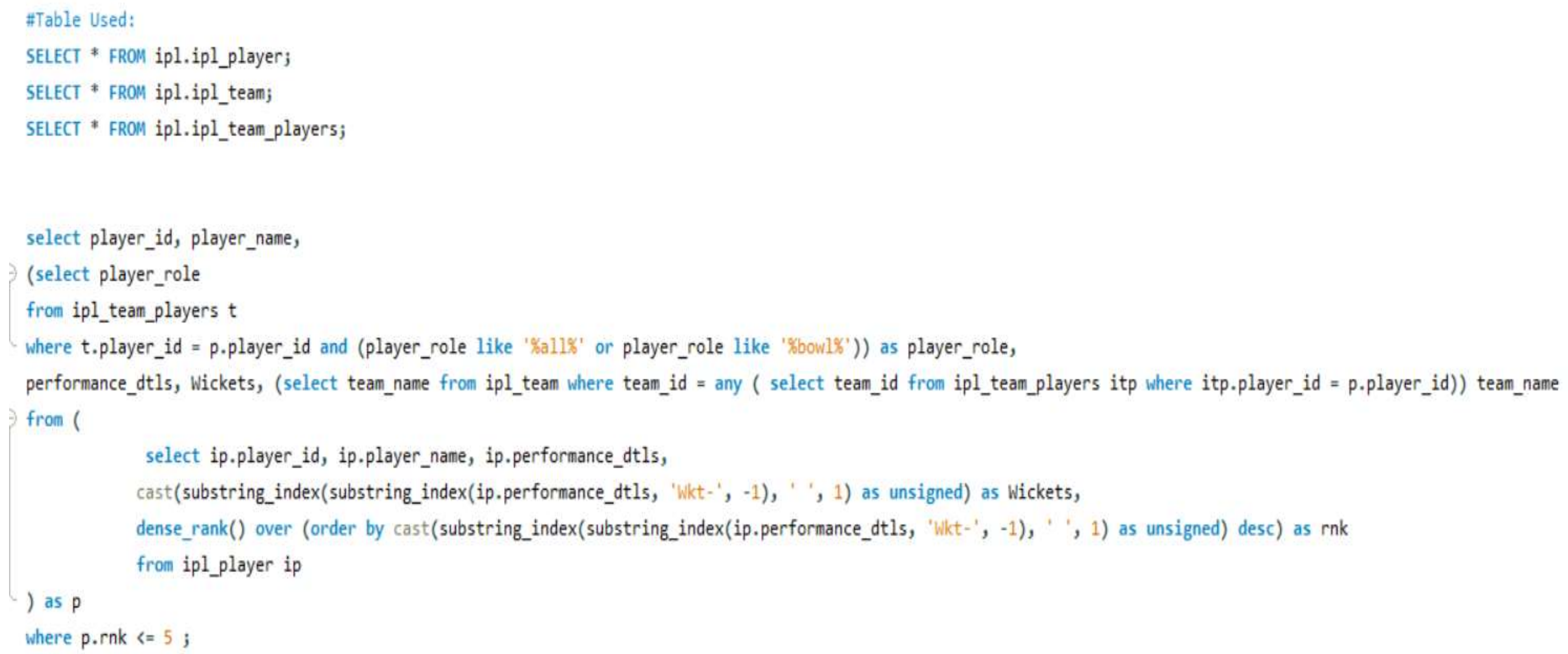
3. Do not use joins in any cases.

4.Display the following columns team_name, player_name, and player_role.

The task is to extract information about the top bowlers and all-rounders based on the highest number of wickets taken

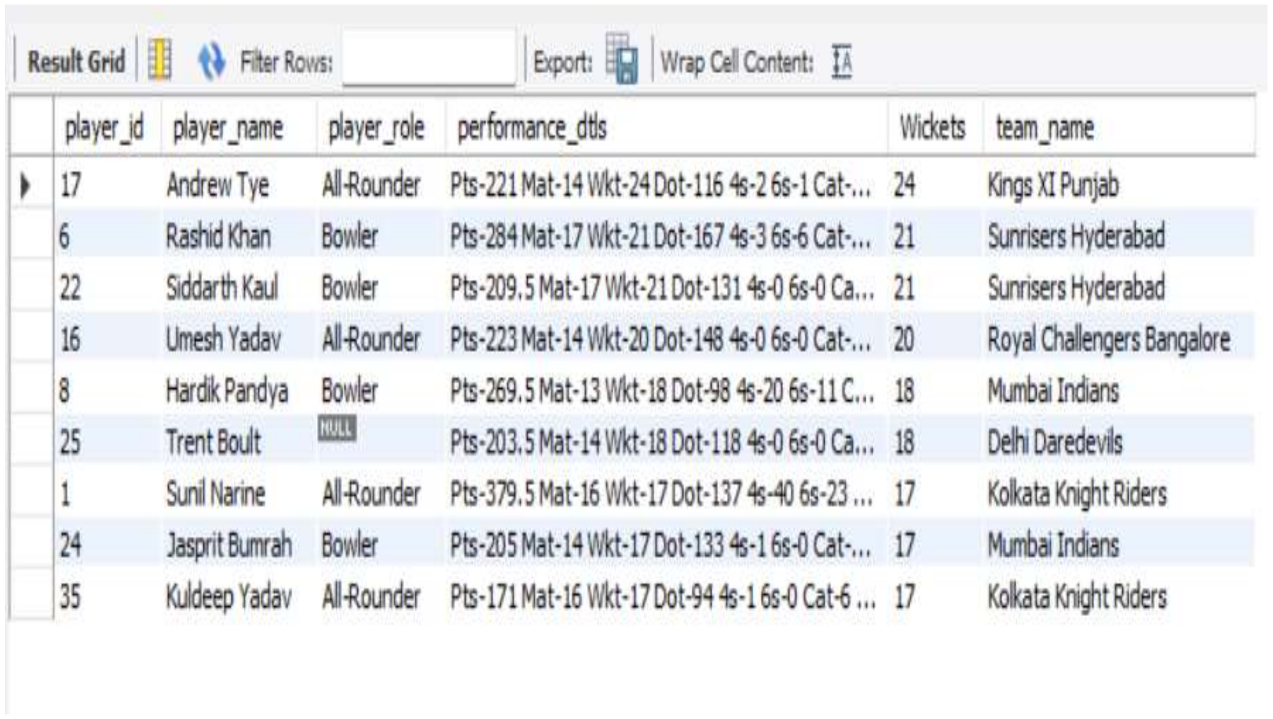
1. **Understand the Data Sources:**I working with a table named ipl_player that contains player details and performance information. The column you need to focus on is performance_dtls
2. **Identify Bowlers and All-Rounders**
3. **Sorting Players by Wickets:** You are instructed not to use the LIMIT method, which suggests that there might be multiple players with the same number of wickets. Hence, you need to sort the players in descending order based on the number of wickets they have taken.
4. **Displaying Specific Columns:** The columns you need to display are team_name, player_name, and player_role.
5. **Avoiding Joins:** You are specifically told not to use joins. This indicates that you won't need to combine data from multiple tables. Instead, you'll be working within the ipl_player table itself.
6. **Execute the Query**

Code:



```
#Table Used:
SELECT * FROM ipl.ipl_player;
SELECT * FROM ipl.ipl_team;
SELECT * FROM ipl.ipl_team_players;


select player_id, player_name,
(select player_role
from ipl_team_players t
where t.player_id = p.player_id and (player_role like '%all%' or player_role like '%bowl%')) as player_role,
performance_dtls, Wickets, (select team_name from ipl_team where team_id = any ( select team_id from ipl_team_players itp where itp.player_id = p.player_id)) team_name
from (
              select ip.player_id, ip.player_name, ip.performance_dtls,
             cast(substring_index(substring_index(ip.performance_dtls, 'Wkt-', -1), ' ', 1) as unsigned) as Wickets,
             dense_rank() over (order by cast(substring_index(substring_index(ip.performance_dtls, 'Wkt-', -1), ' ', 1) as unsigned) desc) as rnk
             from ipl_player ip
) as p
where p.rnk <= 5 ;
```

Output:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

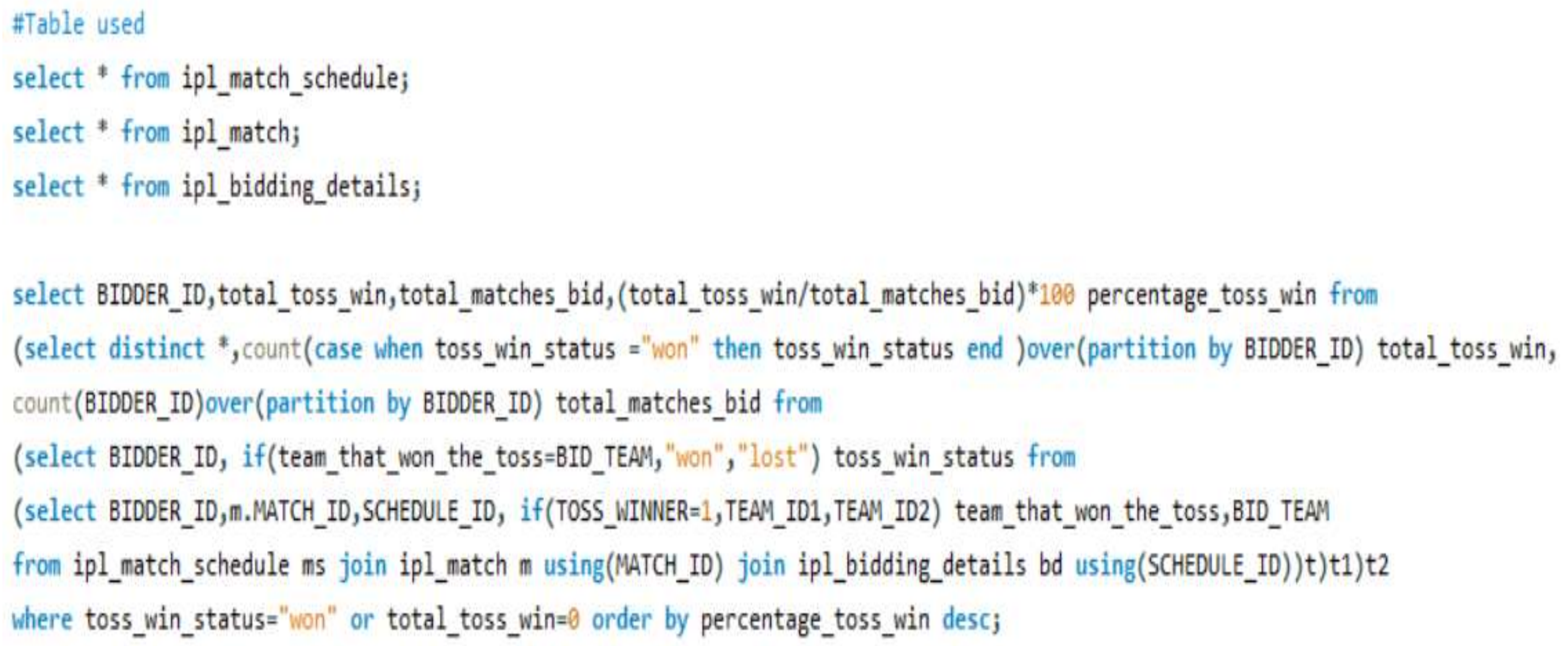
player_id	player_name	player_role	performance_dtls	Wickets	team_name
17	Andrew Tye	All-Rounder	Pts-221 Mat-14 Wkt-24 Dot-116 4s-2 6s-1 Cat-...	24	Kings XI Punjab
6	Rashid Khan	Bowler	Pts-284 Mat-17 Wkt-21 Dot-167 4s-3 6s-6 Cat-...	21	Sunrisers Hyderabad
22	Siddarth Kaul	Bowler	Pts-209.5 Mat-17 Wkt-21 Dot-131 4s-0 6s-0 Ca...	21	Sunrisers Hyderabad
16	Umesh Yadav	All-Rounder	Pts-223 Mat-14 Wkt-20 Dot-148 4s-0 6s-0 Cat-...	20	Royal Challengers Bangalore
8	Hardik Pandya	Bowler	Pts-269.5 Mat-13 Wkt-18 Dot-98 4s-20 6s-11 C...	18	Mumbai Indians
25	Trent Boult	NULL	Pts-203.5 Mat-14 Wkt-18 Dot-118 4s-0 6s-0 Ca...	18	Delhi Daredevils
1	Sunil Narine	All-Rounder	Pts-379.5 Mat-16 Wkt-17 Dot-137 4s-40 6s-23 ...	17	Kolkata Knight Riders
24	Jasprit Bumrah	Bowler	Pts-205 Mat-14 Wkt-17 Dot-133 4s-1 6s-0 Cat-...	17	Mumbai Indians
35	Kuldeep Yadav	All-Rounder	Pts-171 Mat-16 Wkt-17 Dot-94 4s-1 6s-0 Cat-6 ...	17	Kolkata Knight Riders

Q11. show the percentage of toss wins of each bidder and display the results in descending order based on the percentage

The question likely involves analysing the outcomes of coin tosses for these bidders and calculating the percentage of toss wins for each bidder. The goal is to display the results in descending order based on the percentage of toss wins.

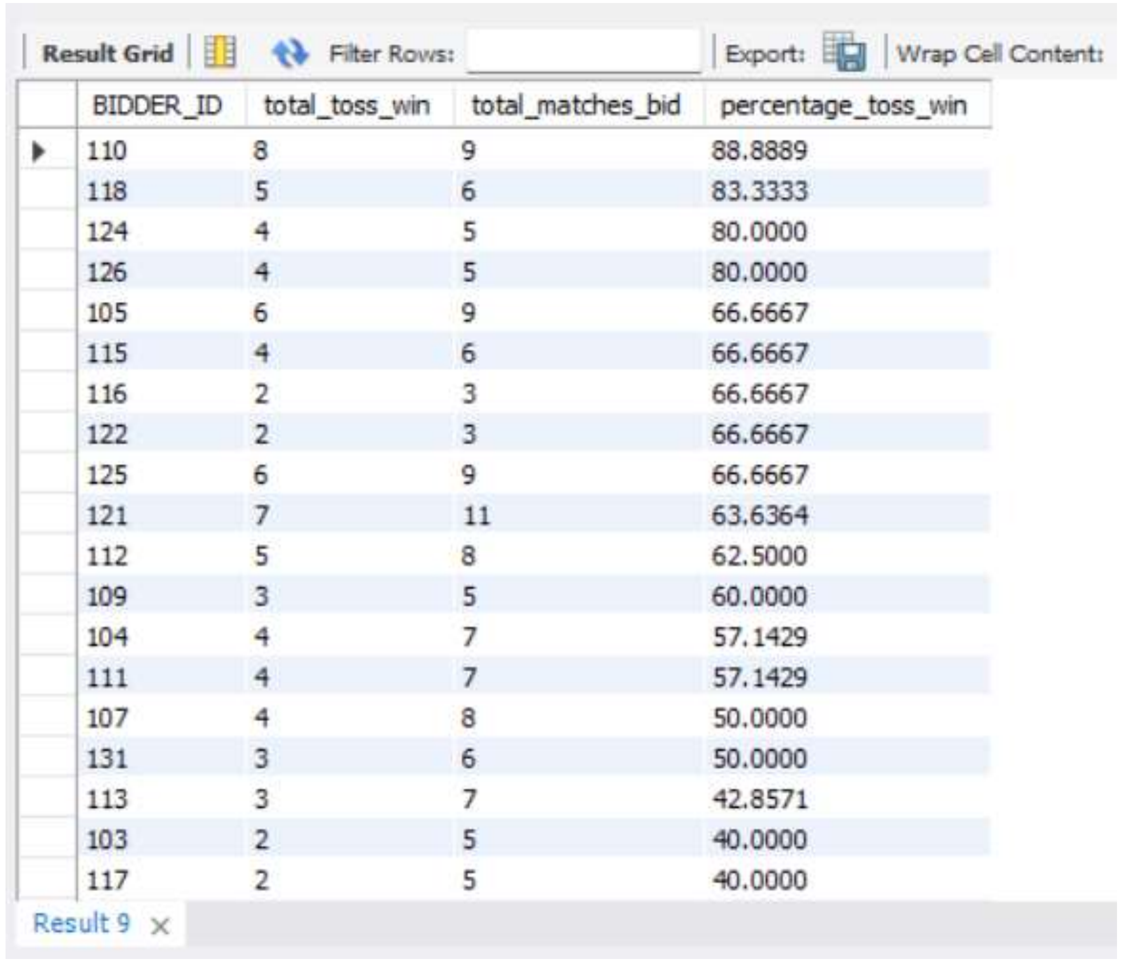
1. Gather Data
2. Calculate Percentages: Percentage of Toss Wins= $\frac{\text{Number of Toss Wins}}{\text{Total Number of Tosses}}$ X 100
3. Arrange in Descending Order
4. Display Results

Code:



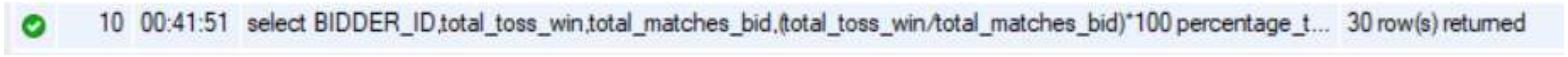
```
#Table used
select * from ipl_match_schedule;
select * from ipl_match;
select * from ipl_bidding_details;

select BIDDER_ID,total_toss_win,total_matches_bid,(total_toss_win/total_matches_bid)*100 percentage_toss_win from
(select distinct *,count(case when toss_win_status ="won" then toss_win_status end )over(partition by BIDDER_ID) total_toss_win,
count(BIDDER_ID)over(partition by BIDDER_ID) total_matches_bid from
(select BIDDER_ID, if(team_that_won_the_toss=BID_TEAM,"won","lost") toss_win_status from
(select BIDDER_ID,m.MATCH_ID,SCHEDULE_ID, if(TOSS_WINNER=1,TEAM_ID1,TEAM_ID2) team_that_won_the_toss,BID_TEAM
from ipl_match_schedule ms join ipl_match m using(MATCH_ID) join ipl_bidding_details bd using(SCHEDULE_ID))t)t1)t2
where toss_win_status="won" or total_toss_win=0 order by percentage_toss_win desc;
```

Output:


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

BIDDER_ID	total_toss_win	total_matches_bid	percentage_toss_win
110	8	9	88.8889
118	5	6	83.3333
124	4	5	80.0000
126	4	5	80.0000
105	6	9	66.6667
115	4	6	66.6667
116	2	3	66.6667
122	2	3	66.6667
125	6	9	66.6667
121	7	11	63.6364
112	5	8	62.5000
109	3	5	60.0000
104	4	7	57.1429
111	4	7	57.1429
107	4	8	50.0000
131	3	6	50.0000
113	3	7	42.8571
103	2	5	40.0000
117	2	5	40.0000

Result 9


```
10  00:41:51  select BIDDER_ID,total_toss_win,total_matches_bid,(total_toss_win/total_matches_bid)*100 percentage_t...  30 row(s) returned
```

Q12. find the IPL season which has min duration and max duration.

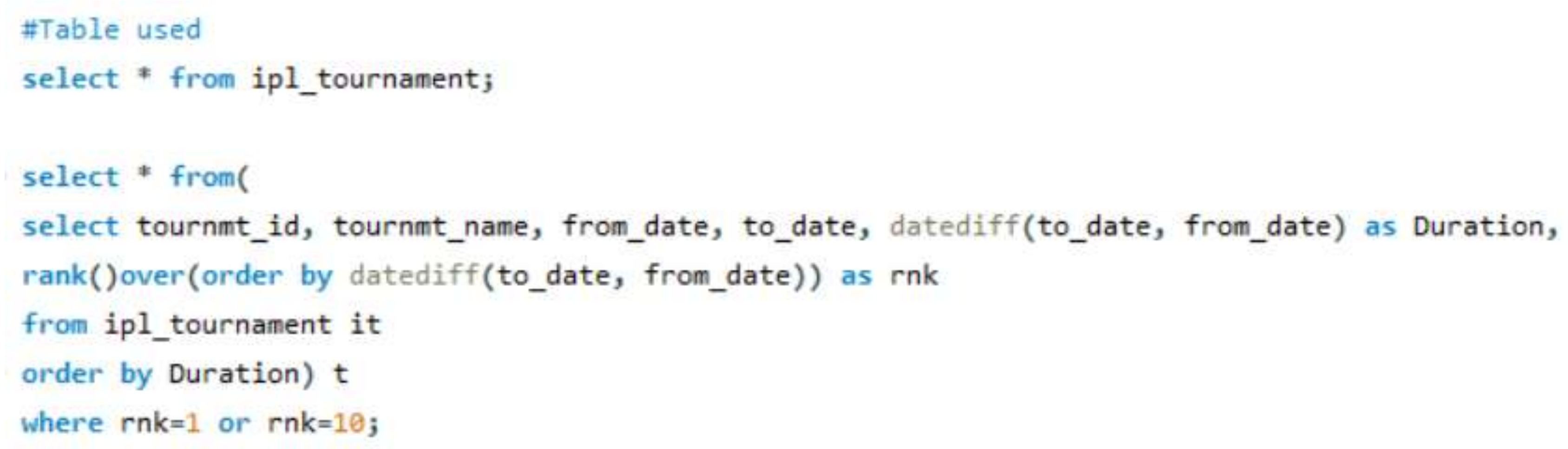
-- Output columns should be like the below:--

Tournment_ID, Tourment_name, Duration column, Duration

The question involves finding the IPL season with the minimum duration and the one with the maximum duration and also required to display specific output columns: Tournament ID, Tournament name, a column indicating whether it's a minimum or maximum duration, and the duration itself.

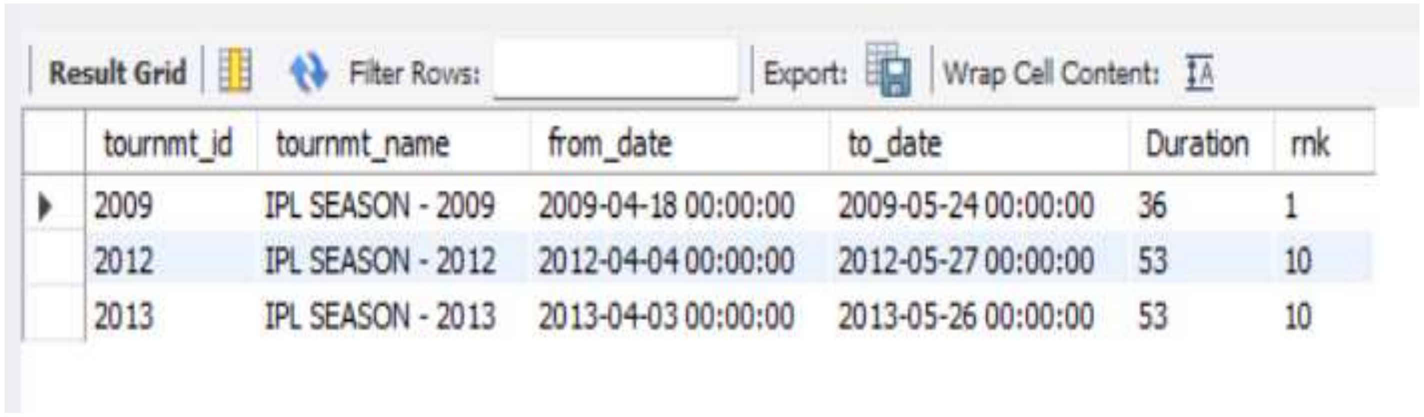
1. Gather Data
2. Calculate Durations
3. Find Min and Max
4. Format Output
5. Display Results

Code:



```
#Table used
select * from ipl_tournament;

select * from(
select tournmt_id, tournmt_name, from_date, to_date, datediff(to_date, from_date) as Duration,
rank()over(order by datediff(to_date, from_date)) as rnk
from ipl_tournament it
order by Duration) t
where rnk=1 or rnk=10;
```

Output:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

tournmt_id	tournmt_name	from_date	to_date	Duration	rnk
2009	IPL SEASON - 2009	2009-04-18 00:00:00	2009-05-24 00:00:00	36	1
2012	IPL SEASON - 2012	2012-04-04 00:00:00	2012-05-27 00:00:00	53	10
2013	IPL SEASON - 2013	2013-04-03 00:00:00	2013-05-26 00:00:00	53	10

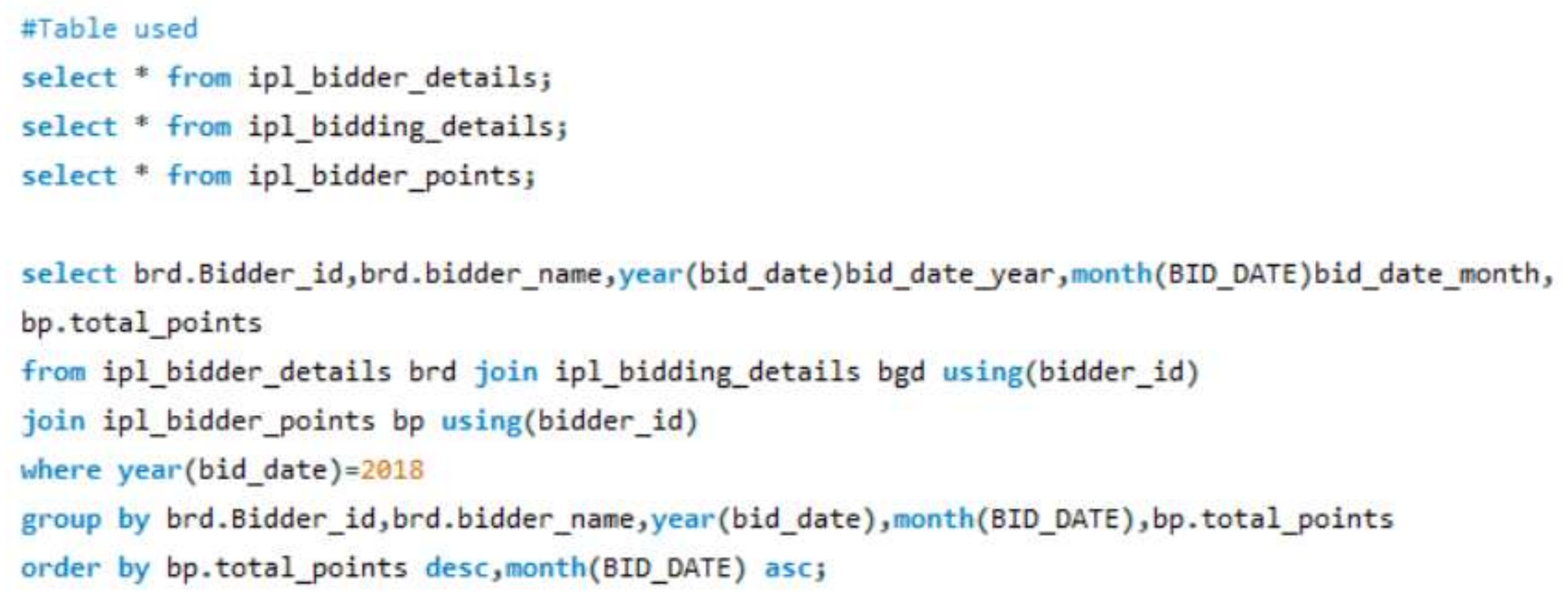
Q 13. Write a query to display to calculate the total points month-wise for the 2017 bid year. sort the results based on total points in descending order and month-wise in ascending order.

-- Note: Display the following columns:--

1. Bidder ID, 2. Bidder Name, 3. bid date as Year, 4. bid date as Month, 5. Total points-- Only use joins for the above query

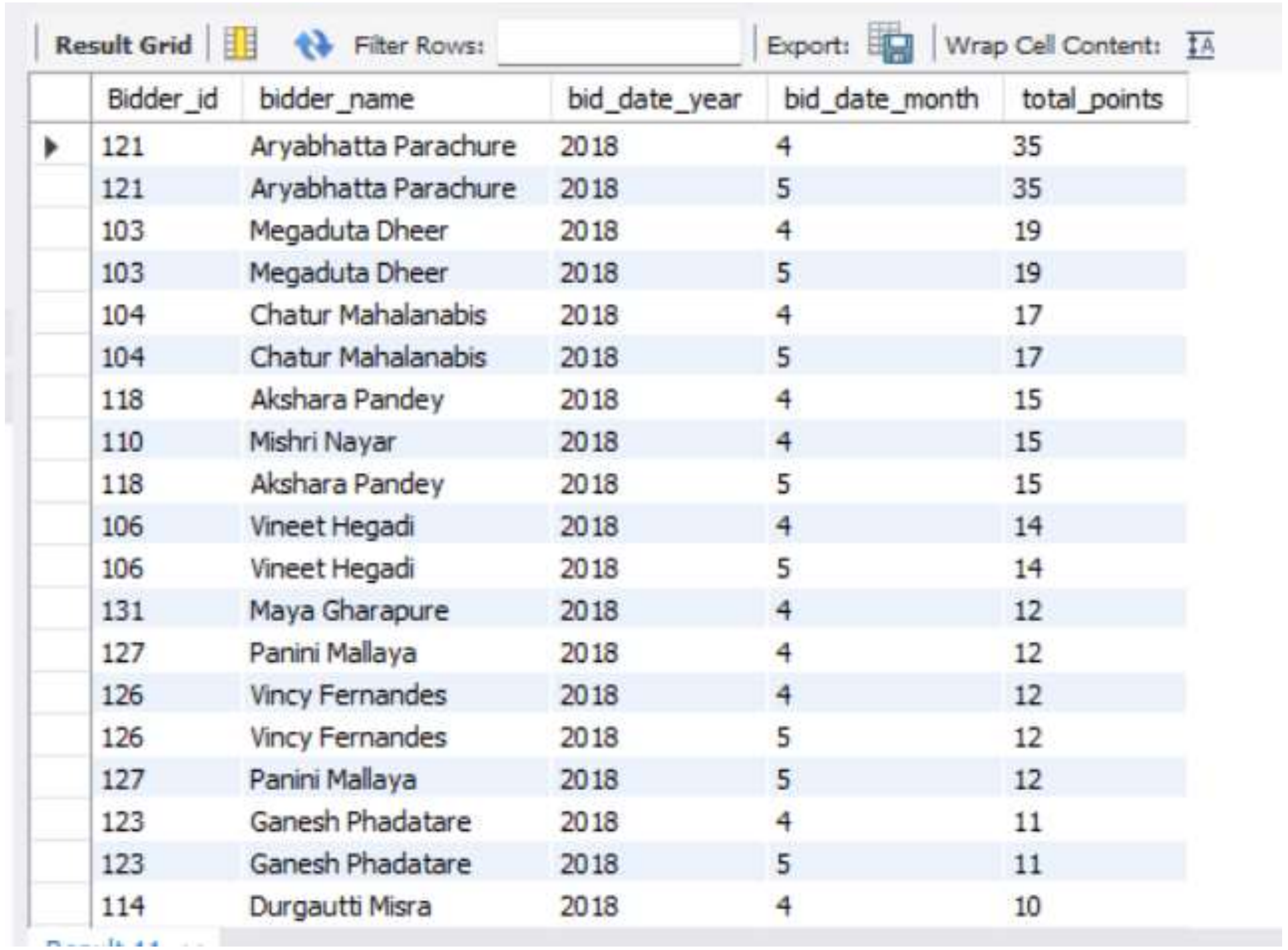
Query to calculate the total points month-wise for the 2017 bid year, and then sorting the results based on total points in descending order and month-wise in ascending order. The query is expected to display specific columns: Bidder ID, Bidder Name, bid date as Year, bid date as Month, and Total points. The constraint is to only use joins for this query.

Code:

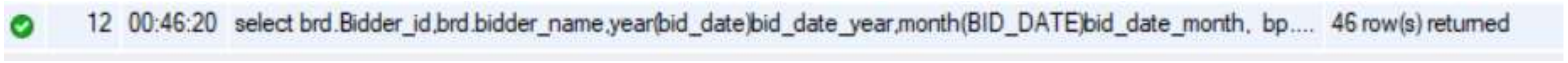


```
#Table used
select * from ipl_bidder_details;
select * from ipl_bidding_details;
select * from ipl_bidder_points;

select brd.Bidder_id,brd.bidder_name,year(bid_date)bid_date_year,month(BID_DATE)bid_date_month,
bp.total_points
from ipl_bidder_details brd join ipl_bidding_details bgd using(bidder_id)
join ipl_bidder_points bp using(bidder_id)
where year(bid_date)=2018
group by brd.Bidder_id,brd.bidder_name,year(bid_date),month(BID_DATE),bp.total_points
order by bp.total_points desc,month(BID_DATE) asc;
```

Output:


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Bidder_id	bidder_name	bid_date_year	bid_date_month	total_points
121	Aryabhatta Parachure	2018	4	35
121	Aryabhatta Parachure	2018	5	35
103	Megaduta Dheer	2018	4	19
103	Megaduta Dheer	2018	5	19
104	Chatur Mahalanabis	2018	4	17
104	Chatur Mahalanabis	2018	5	17
118	Akshara Pandey	2018	4	15
110	Mishri Nayar	2018	4	15
118	Akshara Pandey	2018	5	15
106	Vineet Hegadi	2018	4	14
106	Vineet Hegadi	2018	5	14
131	Maya Gharapure	2018	4	12
127	Panini Mallaya	2018	4	12
126	Vincy Fernandes	2018	4	12
126	Vincy Fernandes	2018	5	12
127	Panini Mallaya	2018	5	12
123	Ganesh Phadatare	2018	4	11
123	Ganesh Phadatare	2018	5	11
114	Durgautti Misra	2018	4	10


```
12  00:46:20  select brd.Bidder_id,brd.bidder_name,year(bid_date)bid_date_year,month(BID_DATE)bid_date_month,  bp....  46 row(s) returned
```

Q14. Write a query to display to calculate the total points month-wise for the 2017 bid year. sort the results based on total points in descending order and month-wise in ascending order.

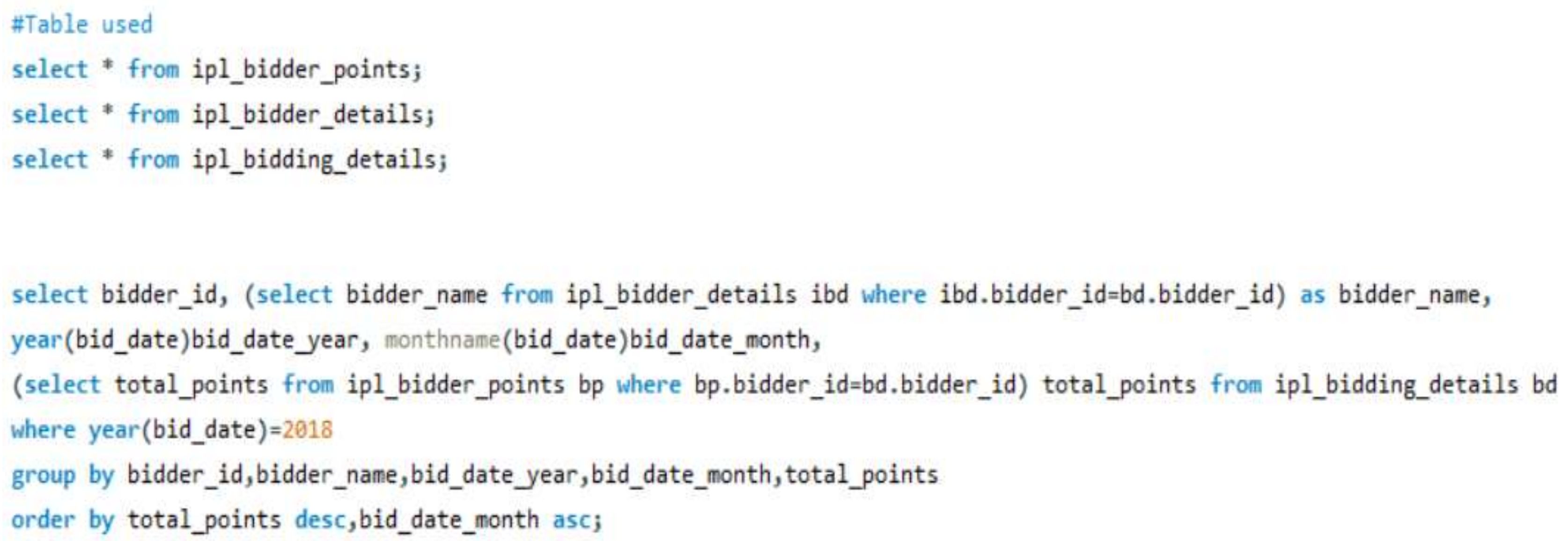
-- Note: Display the following columns:--

1.Bidder ID, 2. Bidder Name, 3. bid date as Year, 4. bid date as Month, 5. Total points-- Don't use joins for the above query .

Query to calculate the total points month-wise for the 2017 bid year, and then sorting the results based on total points in descending order and month-wise in ascending order. The query is expected to display specific columns: Bidder ID, Bidder Name, bid date as Year, bid date as Month, and Total points. The constraint is to only use joins for this query.

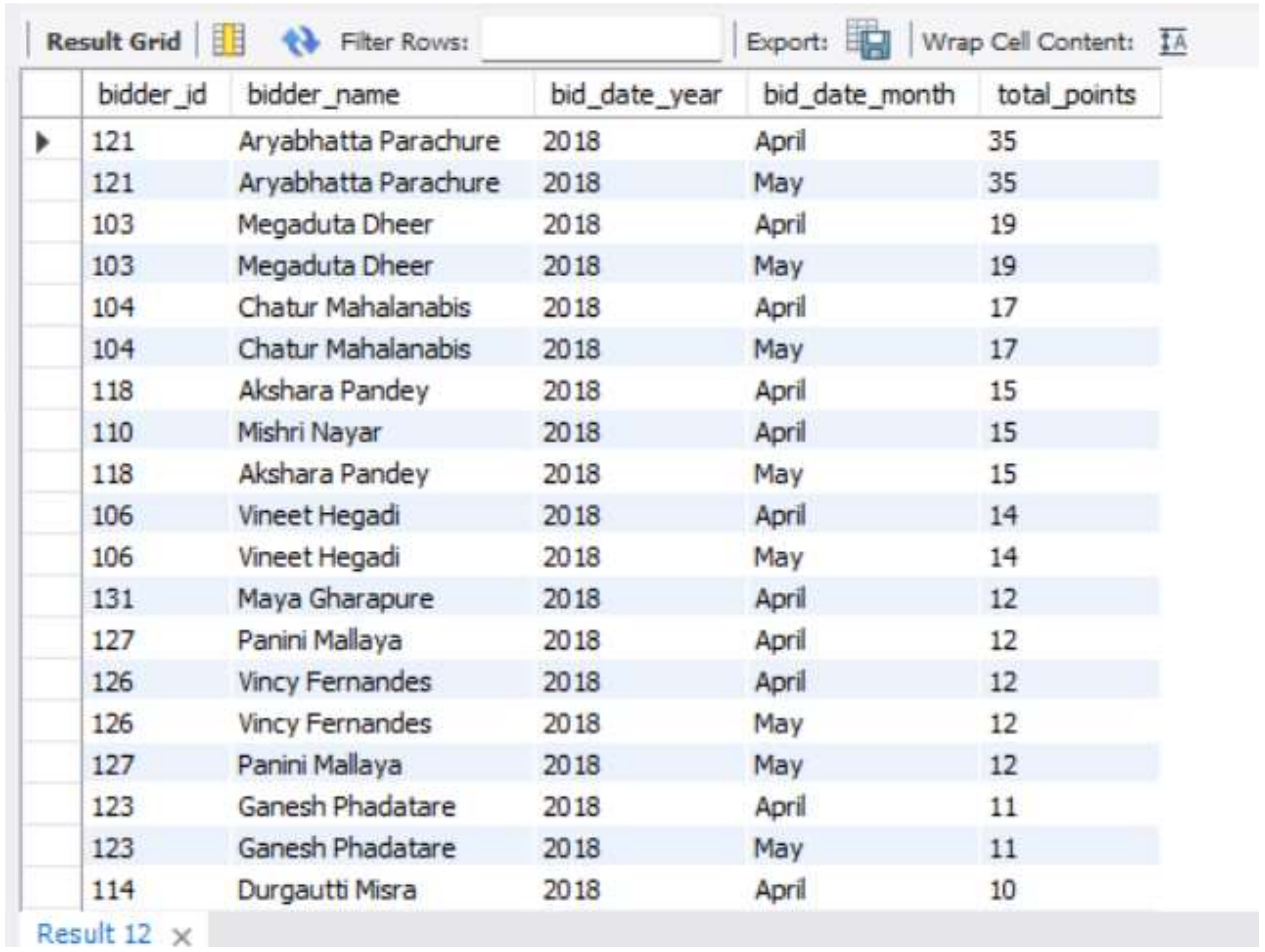
Note: It is same like previous question but in this instead of Joins we used subqueries

Code:

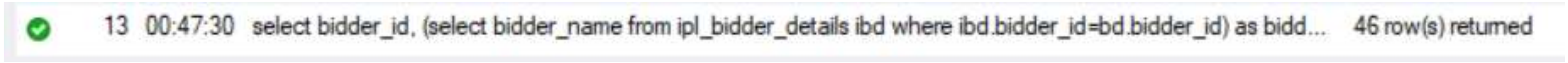


```
#Table used
select * from ipl_bidder_points;
select * from ipl_bidder_details;
select * from ipl_bidding_details;


select bidder_id, (select bidder_name from ipl_bidder_details ibd where ibd.bidder_id=bd.bidder_id) as bidder_name,
year(bid_date)bid_date_year, monthname(bid_date)bid_date_month,
(select total_points from ipl_bidder_points bp where bp.bidder_id=bd.bidder_id) total_points from ipl_bidding_details bd
where year(bid_date)=2018
group by bidder_id,bidder_name,bid_date_year,bid_date_month,total_points
order by total_points desc,bid_date_month asc;
```

Output:



bidder_id	bidder_name	bid_date_year	bid_date_month	total_points
121	Aryabhatta Parachure	2018	April	35
121	Aryabhatta Parachure	2018	May	35
103	Megaduta Dheer	2018	April	19
103	Megaduta Dheer	2018	May	19
104	Chatur Mahalanabis	2018	April	17
104	Chatur Mahalanabis	2018	May	17
118	Akshara Pandey	2018	April	15
110	Mishri Nayar	2018	April	15
118	Akshara Pandey	2018	May	15
106	Vineet Hegadi	2018	April	14
106	Vineet Hegadi	2018	May	14
131	Maya Gharapure	2018	April	12
127	Panini Mallaya	2018	April	12
126	Vincy Fernandes	2018	April	12
126	Vincy Fernandes	2018	May	12
127	Panini Mallaya	2018	May	12
123	Ganesh Phadatare	2018	April	11
123	Ganesh Phadatare	2018	May	11
114	Durgautti Misra	2018	April	10



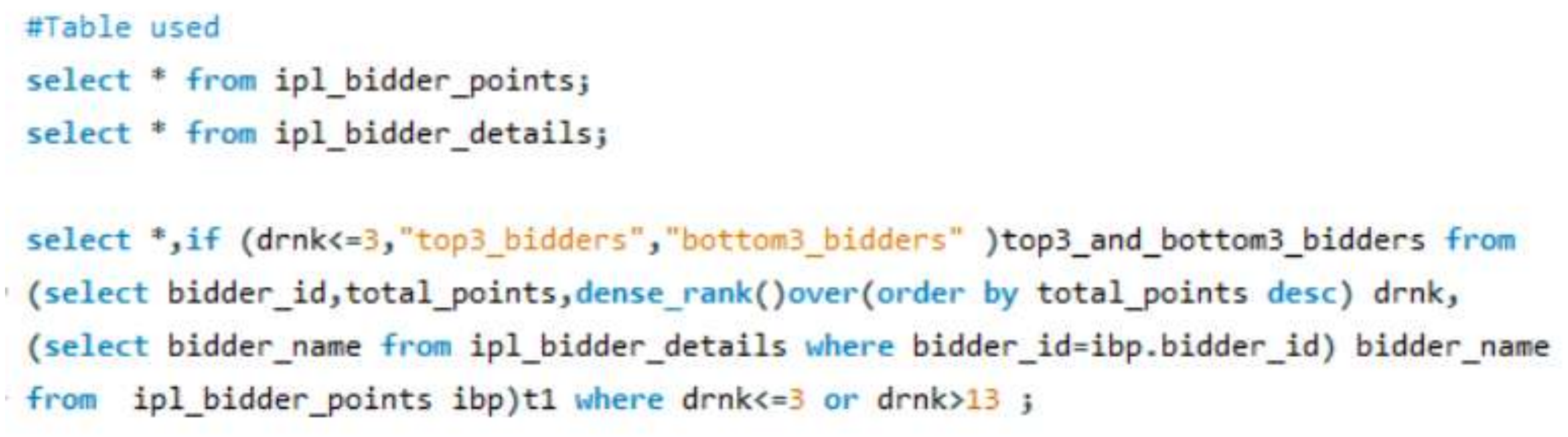
```
13  00:47:30  select bidder_id, (select bidder_name from ipl_bidder_details ibd where ibd.bidder_id=bd.bidder_id) as bidd...    46 row(s) returned
```

Q 15. Write a query to get the top 3 and bottom 3 bidders based on the total bidding points for the 2018 bidding year.

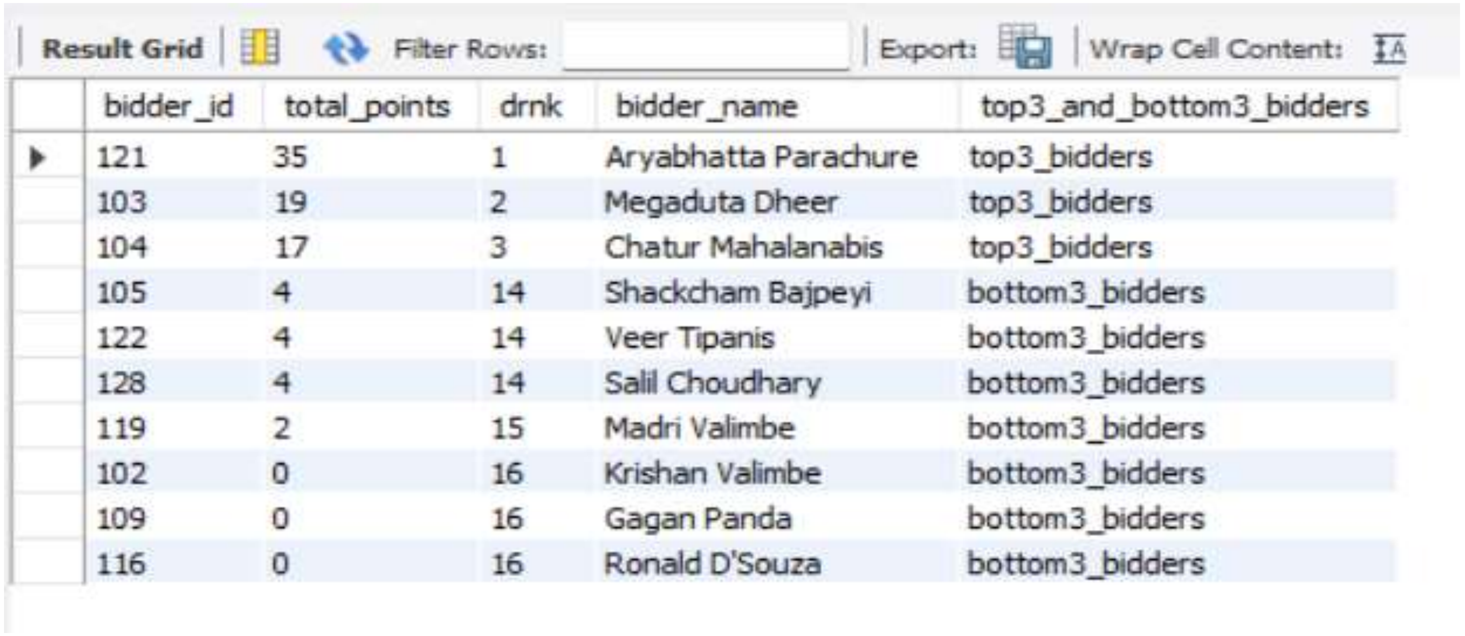
-- Output columns should be like:Bidder Id, Ranks (optional), Total points, Highest_3_Bidders --> columns contains name of bidder, Lowest_3_Bidders -->-- columns contains name of bidder

Query that retrieves information about the top 3 and bottom 3 bidders based on their total bidding points for the year 2018. The query should provide the bidder's ID, their total points, and categorise them into "Highest_3_Bidders" and "Lowest_3_Bidders" based on their ranking in terms of points.

Code:


```
#Table used
select * from ipl_bidder_points;
select * from ipl_bidder_details;

select *,if (drnk<=3,"top3_bidders","bottom3_bidders" )top3_and_bottom3_bidders from
(select bidder_id,total_points,dense_rank()over(order by total_points desc) drnk,
(select bidder_name from ipl_bidder_details where bidder_id=ibp.bidder_id) bidder_name
from  ipl_bidder_points ibp)t1 where drnk<=3 or drnk>13 ;
```

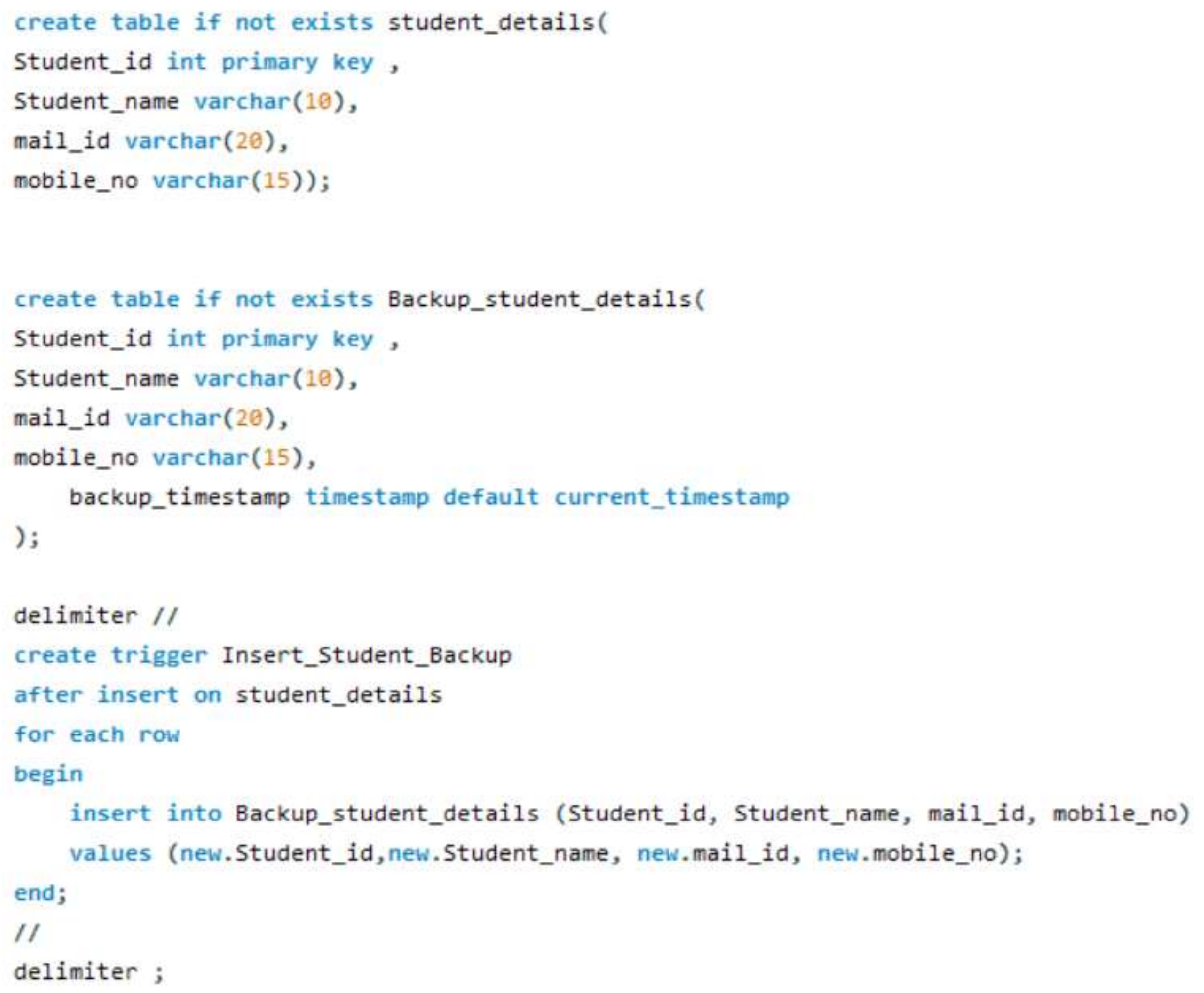
Output:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

bidder_id	total_points	drnk	bidder_name	top3_and_bottom3_bidders
121	35	1	Aryabhatta Parachure	top3_bidders
103	19	2	Megaduta Dheer	top3_bidders
104	17	3	Chatur Mahalanabis	top3_bidders
105	4	14	Shackcham Bajpeyi	bottom3_bidders
122	4	14	Veer Tipanis	bottom3_bidders
128	4	14	Salil Choudhary	bottom3_bidders
119	2	15	Madri Valimbe	bottom3_bidders
102	0	16	Krishan Valimbe	bottom3_bidders
109	0	16	Gagan Panda	bottom3_bidders
116	0	16	Ronald D'Souza	bottom3_bidders

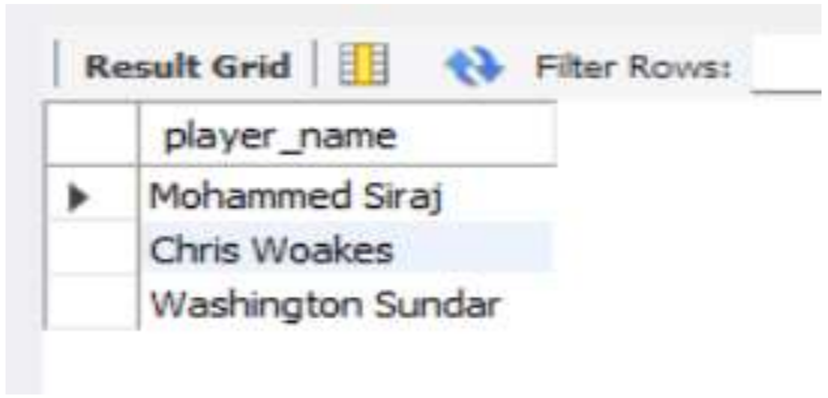
Question16:Create two tables called Student details and Student_details_backup. Create a trigger in such a way that It should insert the details into the Student back table when you inserted the student details into the student table automatically.



```
create table if not exists student_details(
Student_id int primary key ,
Student_name varchar(10),
mail_id varchar(20),
mobile_no varchar(15));


create table if not exists Backup_student_details(
Student_id int primary key ,
Student_name varchar(10),
mail_id varchar(20),
mobile_no varchar(15),
    backup_timestamp timestamp default current_timestamp
);

delimiter //
create trigger Insert_Student_Backup
after insert on student_details
for each row
begin
    insert into Backup_student_details (Student_id, Student_name, mail_id, mobile_no)
    values (new.Student_id,new.Student_name, new.mail_id, new.mobile_no);
end;
//
delimiter ;
```

# 17 . List of RCB Batsmen in the season

```
#Table used
select * from ipl_bidder_points;
select * from ipl_bidder_details;

select player_name from
ipl_player p join ipl_team_players tp
on p.player_id = tp.player_id
join ipl_team t
on tp.team_id = t.team_id
where team_name like "%Royal Challengers Bangalore%" and player_role like "%Batsman%";
```



```
player_name
Mohammed Siraj
Chris Woakes
Washington Sundar
```

Inference : The above query gives the all the players who holds the player role as Batsman in team Royal challengers

Major Challenge

- **Database Design Complexity**
- **Performance Optimization**
- **Time Management**
- SQL projects often involve large datasets and complex queries. Optimizing query performance, indexing, and database configuration can be demanding, especially when dealing with high traffic or resource-intensive operations.

Conclusions

- Lessons learned

1. Data Modelling and Design:

- Lesson: Properly designing the database schema is crucial.
- Details: The schema should accurately represent the entities (e.g., bidders, matches, bids) and their relationships. Consider using appropriate data types, primary keys, foreign keys, and normalization techniques.

2. **Efficient Querying:**

- Lesson: Optimise queries for performance.
- Details: Use indexing on columns frequently used in WHERE clauses or JOIN conditions. Employ query optimisation techniques to reduce response times and resource consumption.

Data Integrity and Validation:

- Lesson: Enforce data integrity through constraints.
- Details: Use constraints like NOT NULL, UNIQUE, FOREIGN KEY, and CHECK to prevent invalid data from entering the database. This ensures reliable and accurate data.

Backup and Recovery:

- Lesson: Regularly back up the database.
- Details: Establish a backup strategy to prevent data loss in case of system failures or disasters. Regularly test the restoration process to ensure backups are effective.

• Skills used

1. Database Design and Modelling:
- Skill: Designing the database schema to represent entities (bidders, matches, bids) and their relationships accurately.
- Explanation: Creating tables, defining primary and foreign keys, and establishing relationships between tables.

2. SQL Querying:
- Skill: Writing SQL queries to retrieve, update, and manipulate data.
- Explanation: Crafting SELECT statements with various clauses (WHERE, JOIN, GROUP BY, ORDER BY) to retrieve specific information from the database.

3. Data Manipulation:
- Skill: Inserting, updating, and deleting data in the database.
- Explanation: Using INSERT, UPDATE, and DELETE statements to manage the content of the database.

4. Data Validation and Constraints:
- Skill: Applying constraints to enforce data integrity and validation rules.
- Explanation: Using NOT NULL, UNIQUE, CHECK, and FOREIGN KEY constraints to prevent invalid data.

5. SQL Joins:
- Skill: Utilizing different types of joins (INNER JOIN, LEFT JOIN, etc.) to retrieve data from multiple related tables.
- Explanation: Understanding how to combine data from different tables based on common keys.

• Domain understanding developed

1. **Understanding IPL Match Bidding**:
    - Domain Knowledge: Gain a thorough understanding of how the IPL match bidding process works, including how teams bid for players, player auctions, match scheduling, team management, and bidding strategies.
    - SQL Application: Design the database schema to accurately model the IPL ecosystem, including entities like teams, players, matches, bids, and their relationships. This understanding informs table structures, relationships, and data flow.

2. **Player Data Management**:
    - Domain Knowledge: Familiarise yourself with player profiles, performance statistics, player transfers, and contractual information relevant to IPL players.
    - SQL Application: Develop database tables that store player data, performance metrics, historical records, and contract details. Design queries to fetch player-specific information for bidding and analysis.

3. **Match Scheduling and Results**:
    - Domain Knowledge: Understand how match schedules are created, how match results are recorded, and how team standings are determined.
    - SQL Application: Build tables to store match schedules, outcomes, and team standings. Create queries to retrieve match-related data, calculate points, and update team standings after each match.

Thank You !!!....