DBMS CASE STUDY ON

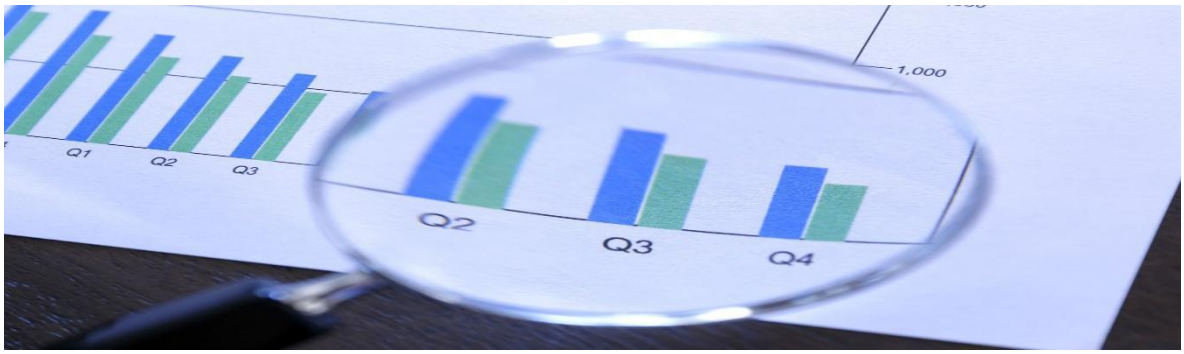
Pie-in-the-Sky

IPL Match Bidding App

APRIL 09,2023

GROUP-1
ONLINE PGP-DSE SEP2022 BATCH

CONTENT



TEAM MEMBERS

- KHUSHBOO PANWAR
- SUBASH
- VARUN

PROBLEM STATEMENT

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. 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 and displays winner. Use SQL queries to explore the given data and write insights based on the results.

TECHNOLOGIES USED IN THIS PROJECT

SQL

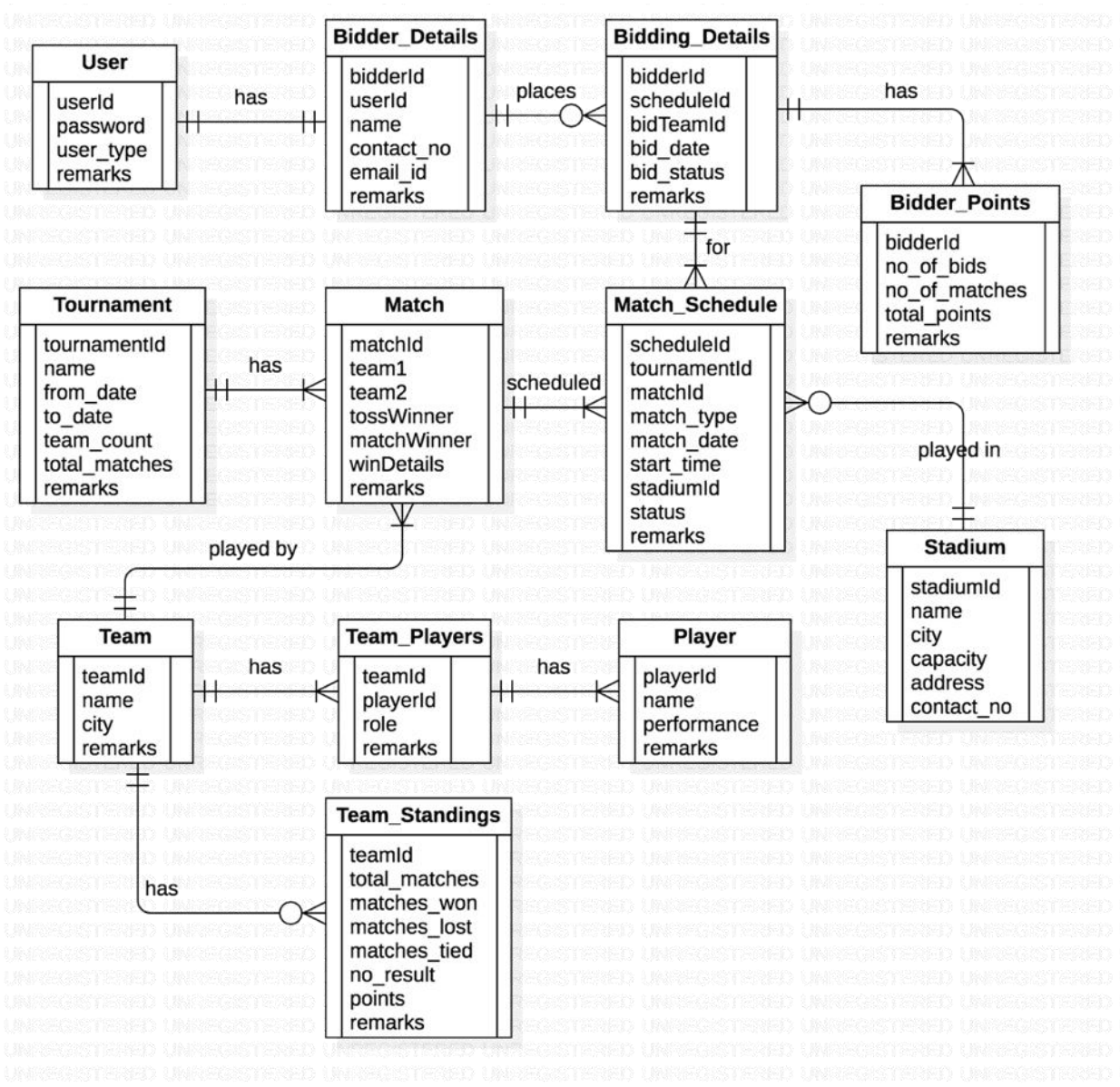
SKILLS USED IN THIS PROJECT

ANALYTICS , PROBLEM SOLVING TECHNIQUES

DATA DESCRIPTION

Table 1- IPL_User (Data Type – VARCHAR)
Table 2- IPL_Stadium (Data Type – VARCHAR , NUMBER)
Table 3- IPL_Team (Data Type – VARCHAR , NUMBER)
Table 4- IPL_Player (Data Type – VARCHAR , NUMBER)
Table 5- IPL_Team_players (Data Type – VARCHAR , NUMBER)
Table 6- IPL_Tournament (Data Type – VARCHAR , NUMBER , DATE)
Table 7- IPL_Match (Data Type – VARCHAR , NUMBER)
Table 8- IPL_Match_Schedule (Data Type – VARCHAR , NUMBER , DATE , TIME)
Table 9- IPL_Bidder_Details (Data Type – VARCHAR , NUMBER)
Table 10- IPL_Bidding_Details (Data Type – VARCHAR , NUMBER , DATE , TIME)
Table 11- IPL_Bidder_Points (Data Type – NUMBER)
Table 12- IPL_Team_Standings (Data Type – VARCHAR , NUMBER)

E-R DIAGRAM



THIS ER DAIGRAM IS THE GRAPHICAL REPRESENTATION OF RELATIONSHIP BETWEEN ENTITIES , ATTRIBUTES , AND USED TO VISUALIZE THE RELATIONSHIP BETWEEN VARIOUS TABLES AND ITS ATTRIBUTES SO AS TO MAKE ANALYSIS EASY TO UNDERSTAND.

DATA EXPLORATION AND INSIGHTS

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

TABLE USED -

Ipl_bidding_details

QUERY -

```
SELECT
   A.BIDDER_ID,ROUND((WIN_COUNT/COUNT(BID_STATUS))*100,2) AS WINS_PERCENTAGE
FROM
   IPL_BIDDING_DETAILS A
JOIN

(SELECT BIDDER_ID , COUNT(BID_STATUS) AS WIN_COUNT

FROM
   IPL_BIDDING_DETAILS B WHERE BID_STATUS = 'WON' GROUP BY BIDDER_ID)T
ON A.BIDDER_ID = T.BIDDER_ID
GROUP BY
   A.BIDDER_ID ORDER BY WINS_PERCENTAGE DESC ;
```

OUTCOME - 27 ROWS

Result Grid | Filter Rows:

Bidder_id	wins_percentage
103	100.00
121	90.91
118	83.33
126	80.00
104	71.43
122	66.67

CONCLUSION -

In above question we concluded that Bidder with Bidder ID 103 has highest winning percentage i.e 100% followed by bidder ID 121 with winning percentage 90.91% and bidder id 19 has the lowest winning percentage 10%

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

TABLES USED -

ipl_stadium, ipl_match_schedule

QUERY -

```
SELECT
   a.STADIUM_ID,
   a.STADIUM_NAME,
   a.City,
   COUNT(match_id) AS No_of_matches
FROM
   ipl_stadium a
      JOIN
   ipl_match_schedule b ON a.STADIUM_ID = b.STADIUM_ID
GROUP BY a.STADIUM_ID
ORDER BY a.STADIUM_ID;
```

OUTCOME - 10 ROWS

	STADIUM_ID	STADIUM_NAME	City	COUNT(match_id)
▶	1	Wankhede Stadium	Mumbai	18
	2	Feroz Shah Kotla	Delhi	13
	3	Eden Gardens	Kolkata	13
	4	Rajiv Gandhi International Stadium	Hyderabad	7
	5	MS Chidambaram Stadium	Chennai	12
	6	Sawai Mansingh Stadium	Jaipur	10
	7	M. Chinnaswamy Stadium	Bengaluru	13
	8	Is Bindra Stadium	Mohali	16
	9	Holkar Stadium	Indore	13
	10	MCA Stadium	Pune	7

CONCLUSION -

In above outcome we analysed that highest no of matches were conducted at wankhede stadium , Mumbai followed by Is Bindra stadium, Mohali and the least No of matches conducted at Rajiv gandhi and MCA stadium.

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

TABLE USED -

ipl_match

QUERY -

```
SELECT SUM(
  CASE
    WHEN
          ( toss_winner = match_winner )
          THEN 1
          ELSE 0
  END) / COUNT(*) * 100 "Percentage of wins by a team which has won the toss"
FROM ipl_match;
```

OUTCOME - 1 ROW

	Percentage of wins by a team which has won the toss
▶	46.6667

CONCLUSION -

##In above scenerio, we concluded that the Percentage of wins by a team which has won the toss is 46.6667.

Q4.Show the total bids along with the bid team and team name.

TABLES USED -

ipl_bidder_points
ipl_bidding_details
Ipl_team

QUERY -

```
SELECT
   b.bid_team, c.TEAM_NAME, SUM(a.no_of_bids) AS total_bids
FROM
   ipl_bidder_points a
      JOIN
   ipl_bidding_details b ON a.BIDDER_ID = b.BIDDER_ID
      JOIN
   ipl_team c ON b.BID_TEAM = c.TEAM_ID
GROUP BY b.bid_team
ORDER BY b.bid_team;
```

OUTCOME - 8 ROWS

bid_team	TEAM_NAME	total_bids
1	Chennai Super Kings	162
2	Delhi Daredevils	190
3	Kings XI Punjab	182
4	Kolkata Knight Riders	164
5	Mumbai Indians	157
6	Rajasthan Royals	211
7	Royal Challengers Bangalore	171
8	Sunrisers Hyderabad	227

CONCLUSION -

##Above table is showing us the count of total bids of each team where sunrisers Hyderabad has highest total bids and Mumbai indians has lowest total bids.

Q5.Show the team id who won the match as per the win details.

TABLES USED -

ipl_match

QUERY -

```
SELECT
   match_id,
   WIN_DETAILS,
   CASE
      WHEN match_winner = 1 THEN team_id1
      WHEN match_winner = 2 THEN team_id2
      ELSE 0
   END AS winning_team_id
FROM
   ipl_match;
```

OUTCOME - 100 ROWS

match_id	WIN_DETAILS	winning_team_id
1001	Team CSK won by 7 Wkts	1
1002	Team CSK won by 7 Wkts	1
1003	Team KKR won by 35 Runs	4
1004	Team CSK won by 7 Wkts	1
1005	Team RR won by 35 Runs	6
1006	Team RCB won by 35 Runs	7
1007	Team DD won by 35 Runs	2
1008	Team DD won by 35 Runs	2
1009	Team DD won by 35 Runs	2
1010	Team MI won by 7 Wkts	5

CONCLUSION -

##Using CASE WHEN here in this query we have got the desired outcome that which team has who won the match as per the win details.

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

TABLES USED -

Ipl_team
ipl_team_standings

QUERY -

```
 SELECT
    a.team_name,
    SUM(matches_played) total_matches_played,
    SUM(matches_won) total_matches_won,
    SUM(matches_lost) total_matches_lost
FROM
    ipl_team a
        JOIN
    ipl_team_standings b ON a.team_id = b.team_id
GROUP BY a.team_name;
```

OUTCOME - 8 ROWS

team_name	total_matches_played	total_matches_won	total_matches_lost
Chennai Super Kings	28	18	10
Delhi Daredevils	28	11	17
Kings XI Punjab	28	13	15
Kolkata Knight Riders	28	16	12
Mumbai Indians	28	16	12
Rajasthan Royals	28	16	12
Royal Challengers Bangalore	28	9	18
Sunrisers Hyderabad	28	17	10

CONCLUSION -

As asked in question here we have summarised total matches played , total matches won and total matches lost by each team

Q7.Display the bowlers for the Mumbai Indians team.

TABLES USED -

Ipl_player
Ipl_team_players
Ipl_team

QUERY -

```
SELECT
   a.player_name, c.Team_name, b.player_role
FROM
   ipl_player a
      JOIN
   ipl_team_players b ON a.PLAYER_ID = b.PLAYER_ID
      JOIN
   ipl_team c ON b.team_id = c.team_id
WHERE
   c.team_name LIKE '%Mumbai Indians%'
      AND b.player_role LIKE '%Bowler%';
```

OUTCOME - 9 ROWS

player_name	Team_name	player_role
Hardik Pandya	Mumbai Indians	Bowler
Suryakumar Yadav	Mumbai Indians	Bowler
Jasprit Bumrah	Mumbai Indians	Bowler
Evin Lewis	Mumbai Indians	Bowler
Mayank Markande	Mumbai Indians	Bowler
Rohit Sharma	Mumbai Indians	Bowler
Ben Cutting	Mumbai Indians	Bowler
Kieron Pollard	Mumbai Indians	Bowler
JP Duminy	Mumbai Indians	Bowler

CONCLUSION -

Here with the help of joins we have extracted the bowlers from Mumbai Indians and we came to know that Mumbai Indians has total 9 bowlers.

Q8.How many all-rounders are there in each team, Display the teams with more than 4 all-rounders in descending order.

TABLES USED -

Ipl_team
Ipl_team_players

QUERY -

```
SELECT
   a.team_name, COUNT(player_role) AS All_rounders
FROM
   ipl_team a
      JOIN
   ipl_team_players b ON a.team_id = b.team_id
WHERE
   player_role = 'All-Rounder'
GROUP BY a.team_name
HAVING All_rounders > 4
ORDER BY All_rounders DESC;
```

OUTCOME - 5 ROWS

team_name	All_rounders
Delhi Daredevils	7
Kings XI Punjab	7
Sunrisers Hyderabad	6
Kolkata Knight Riders	5
Rajasthan Royals	5

CONCLUSION -

##Here we have extracted the teams with more than 4 all-rounders and we came to know that Delhi Daredevils has more no of all rounders.

Q9.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.

TABLES USED -

Ipl_team , ipl_match_schedule , ipl_match , ipl_stadium , ipl_bidding_details

QUERY -

```
SELECT a.BIDDER_ID,YEAR(bid_date),a.bid_status,SUM(b.TOTAL_POINTS) FROM
ipl_bidding_details a JOIN ipl_bidder_points b ON a.BIDDER_ID = b.BIDDER_ID
GROUP BY a.BIDDER_ID , a.bid_status , YEAR(bid_date)
HAVING bidder_id IN ((SELECT bidder_id FROM ipl_bidding_details WHERE schedule_id IN
(SELECT ad.schedule_id FROM(SELECT stadium_name, stadium_id FROM ipl_stadium) a
JOIN (SELECT T1.match_id, stadium_id, winning_team_id, schedule_id FROM
(SELECT match_id, Stadium_id, schedule_id FROM ipl_match_schedule) T1
JOIN (SELECT match_id,
CASE WHEN match_winner = 1 THEN team_id1
        WHEN match_winner = 2 THEN team_id2
        ELSE 0
        END AS winning_team_id  FROM ipl_match WHERE Win_details LIKE '%CSK%') T2 ON
T1.match_id = T2.match_id AND stadium_id = 7) ad ON ad.stadium_id = a.stadium_id)))
   ORDER BY SUM(b.TOTAL_POINTS) DESC;
```

OUTCOME - 7 ROWS

Result Grid | Filter Rows: | Export: | W

BIDDER_ID	year(bid_date)	bid_status	sum(b.TOTAL_POINTS)
104	2017	Won	68
104	2017	Cancelled	17
104	2018	Cancelled	17
104	2018	Won	17
102	2017	Bid	0
102	2017	Lost	0
102	2018	Lost	0

CONCLUSION –

Above result shows that bidder id 104 has the maximum total points when CSK won the match in M. Chinnaswamy Stadium.

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 teamn_name, player_name, and player_role.

TABLES USED –

Ipl_team , ipl_team_players, Ipl_player , ipl_stadium , ipl_bidding_details

QUERY –

```
SELECT team "Team name" , player_name "Player name" , role "Player role" , wickets "Wickets"
FROM( SELECT * , DENSE_RANK() OVER(ORDER BY wickets DESC) "rank"
FROM( SELECT * ,
( SELECT team_name FROM ipl_team WHERE team_id=t3.team_id ) "team" FROM
( SELECT player_name ,
( SELECT team_id FROM ipl_team_players WHERE t2.player_id=player_id) "team_id",
( SELECT player_role FROM ipl_team_players WHERE t2.player_id=player_id) "role",
CONVERT(SUBSTR(wkt,1,LENGTH(wkt)-LENGTH(substr(wkt,INSTR(wkt," )))) , FLOAT) "wickets"
FROM
( SELECT *,SUBSTR(wkts,INSTR(wkts,"-")+1) "wkt" from
( SELECT *,SUBSTR(performance_dtls,INSTR(performance_dtls,"W")) "wkts" from
( SELECT * from Ipl_player)t )t1 )t2
ORDER BY
Wickets DESC)t3 )t4 )t5
WHERE
`rank`<=5 AND (`role`="Bowler" OR `role`="All-Rounder");
```

OUTCOME - 8 ROWS

Result Grid | Filter Rows: | Export: | Wrap

Team name	Player name	Player role	Wickets
Kings XI Punjab	Andrew Tye	All-Rounder	24
Sunrisers Hyderabad	Rashid Khan	Bowler	21
Sunrisers Hyderabad	Siddarth Kaul	Bowler	21
Royal Challengers Bangalore	Umesh Yadav	All-Rounder	20
Mumbai Indians	Hardik Pandya	Bowler	18
Kolkata Knight Riders	Sunil Narine	All-Rounder	17
Mumbai Indians	Jasprit Bumrah	Bowler	17
Kolkata Knight Riders	Kuldeep Yadav	All-Rounder	17

CONCLUSION –

THERE ARE 8 PLAYERS INCLUDING BOWLERS AND ALL-ROUNDERS WHO SECURED PLACE IN TOP 5 HIGHEST NUMDER OF WICKETS TAKEN PALYERS IN WHICH ANDREW TYRE OF KINGS XI PUNJAB EARNED HIGHEST WICKETS.

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

TABLES USED –

Ipl_team , ipl_match, Ipl_player , ipl_bidding_details

QUERY –

```
select
    dd.bidder_id , dd.total_toss_wins, round((dd.total_toss_wins*100/(select count(toss_winner)
from
    ipl_match)),2) as percentage_toss_wins
from(select T1.bidder_id , count(T2.winning_toss_ID) as total_toss_wins
from(select bidder_id , bid_team from ipl_bidding_details)T1
join
     (select
     CASE
       WHEN toss_winner =1  THEN team_id1
       when toss_winner =2 then team_id2
       ELSE 0
       END AS winning_toss_ID
from
    ipl_match where toss_winner)T2
    on T1.bid_team=T2.winning_toss_ID
group by T1.bidder_id)dd
       group by dd.bidder_id , dd.total_toss_wins
order by percentage_toss_wins desc;
```

OUTCOME - 30 ROWS

Result Grid | Filter Rows:

bidder_id	total_toss_wins	percentage_toss_wins
121	165	137.50
119	153	127.50
106	149	124.17
110	142	118.33
105	138	115.00
125	132	110.00

CONCLUSION –

Here from above outcome we concluded that bidder with bidder id 121 won the highest percentage of toss wins.

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

TABLES USED –

Ipl_team , ipl_match, Ipl_player , ipl_bidding_details

QUERY –

```
select Tournmt_id,tournmt_name, Duration_days,
 case
 when Duration_days =(select  max(datediff(To_date,from_date)) from ipl_tournament) then 'Max'
 when Duration_days =(select  min(datediff(To_date,from_date)) from ipl_tournament) then 'Min'
 else 0
end as Duration from
 (select Tournmt_id,tournmt_name,datediff(To_date,from_date) as Duration_days  from
ipl_tournament
where datediff(To_date,from_date) = (select  max(datediff(To_date,from_date)) from ipl_tournament)
or
 datediff(To_date,from_date) = (select  min(datediff(To_date,from_date)) from ipl_tournament))T ;
```

OUTCOME - 30 ROWS

Result Grid | Filter Rows: | Export:

Tournmt_id	tournmt_name	Duration_days	Duration
2009	IPL SEASON - 2009	36	Min
2012	IPL SEASON - 2012	53	Max
2013	IPL SEASON - 2013	53	Max

CONCLUSION –

It is concluded that maximum duration IPL SEASON was 2012 , 2013 and IPL season with minimum duration is 2009.

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 queries.

TABLES USED –

ipl_bidder_points, ipl_bidder_details , ipl_bidding_details

QUERY –

```
select
     T1.bidder_id , T1.bidder_Name, month(T2.bid_date) as bid_month , year(T2.bid_date) as
bid_year ,sum(T3.total_points) as total_points
from
     ipl_bidder_details as T1 left join ipl_bidding_details as T2 on T1.bidder_id= T2.bidder_id
 join
      ipl_bidder_points as T3
 on T1.bidder_id = T3.bidder_id where year(T2.bid_date) = '2017'
group by
      T1.bidder_id,T1.bidder_name, year(T2.bid_date) ,month(T2.bid_date)
 order by
      month(T2.bid_date), sum(T3.total_points) desc;
```

OUTCOME - 55 ROWS

Result Grid | Filter Rows: | Export: | Wrap Ce

bidder_id	bidder_Name	bid_month	bid_year	total_points
121	Aryabhatta Parachure	4	2017	105
129	Aryabhata Valimbe	4	2017	45
110	Mishri Nayar	4	2017	45
104	Chatur Mahalanabis	4	2017	34
125	Gagan Adwani	4	2017	32
118	Akshara Pandey	4	2017	30
106	Vineet Hegadi	4	2017	28

CONCLUSION –

From the outcome it shows the total points monthwise of each bidder for the year 2017 in which bidder named Aryabhatta Parachure has the maximum total points

and Ronald D'souza,Gagan panda has not won any bid.

Q14.Write a query for the above question using sub queries by having the same constraints as the above question.

TABLES USED –

ipl_bidder_details ,ipl_bidding_details ,ipl_bidder_points

QUERY –

```
with T as
(select a.bidder_id,a.bidder_name,year(b.bid_date) as Year ,month(b.bid_date) as
Month ,sum(c.total_points) as Total_points
from ipl_bidder_details a,ipl_bidding_details b,ipl_bidder_points c
where a.BIDDER_ID = b.BIDDER_ID and b.BIDDER_ID=c.BIDDER_ID
group by a.bidder_id,a.bidder_name,year(b.bid_date),month(b.bid_date) order by
month(b.bid_date),sum(c.total_points) desc)
select * from T where Year ='2017';
```

OUTCOME - 55 ROWS

bidder_id	bidder_name	Year	Month	Total_points
121	Aryabhatta Parachure	2017	4	105
129	Aryabhata Valimbe	2017	4	45
110	Mishri Nayar	2017	4	45
104	Chatur Mahalanabis	2017	4	34
125	Gagan Adwani	2017	4	32
118	Akshara Pandey	2017	4	30
106	Vineet Hegadi	2017	4	28
124	Sackhcham Nayar	2017	4	21
103	Megaduta Dheer	2017	4	19
112	Shinu Sanyal	2017	4	18

CONCLUSION –

From the outcome it shows the total points monthwise of each bidder for the year 2017 in which bidder named Aryabhatta Parachure has the maximum total points and Ronald D'souza,Gagan panda has not won any bid.

Q15.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;

TABLES USED –
ipl_bidder_details ,ipl_bidding_details ,ipl_bidder_points

QUERY –
```
SELECT rank_id "Rank",`Lowest bidder ID`,`Lowest bidder`,points,`Highest bidder id`,`Highest
bidder`,total_points"Points"
FROM ( SELECT * FROM( SELECT * , ROW_NUMBER() OVER(ORDER BY points) "Rank_id"
FROM( SELECT a.bidder_id "Lowest bidder id", bidder_name "Lowest bidder" , YEAR(b.bid_date) ,
SUM(c.TOTAL_POINTS) AS "Points"  FROM ipl_bidder_details a JOIN ipl_bidding_details b ON
a.BIDDER_ID=b.BIDDER_ID JOIN ipl_bidder_points c ON b.BIDDER_ID=c.BIDDER_ID
 WHERE YEAR(b.bid_date)='2018' GROUP BY a.BIDDER_ID , a.BIDDER_NAME , YEAR(b.bid_date)
 ORDER BY SUM(c.TOTAL_POINTS) LIMIT 3)t1)t2
 JOIN ( SELECT * FROM( SELECT *  , ROW_NUMBER() OVER(ORDER BY total_points DESC)
"Rank_id"FROM( SELECT a.bidder_id "Highest bidder Id",a.Bidder_name "Highest
bidder",SUM(c.TOTAL_POINTS) "total_points"
  FROM ipl_bidder_details a
  JOIN ipl_bidding_details b ON a.BIDDER_ID=b.BIDDER_ID
  JOIN ipl_bidder_points c ON b.BIDDER_ID=c.BIDDER_ID
  WHERE YEAR(b.bid_date)='2018'
  GROUP BY a.BIDDER_ID , a.BIDDER_NAME , YEAR(b.bid_date) , c.TOTAL_POINTS
  ORDER BY SUM(c.TOTAL_POINTS) DESC LIMIT 3)t3)t4)t5
 USING(rank_id))t6;
```

OUTCOME - 3 ROWS

	Rank	Lowest bidder id	Lowest bidder	Points	Highest bidder Id	Highest bidder	Points
▶	1	102	Krishan Valimbe	0	121	Aryabhatta Parachure	210
	2	116	Ronald D'Souza	0	110	Mishri Nayar	75
	3	109	Gagan Panda	0	106	Vineet Hegadi	70

CONCLUSION –Above result shows the top three and bottom three bidders based on the total points for the year 2018.

Q16.Create two tables called Student_details and Student_details_backup.

Table 1: Attributes Table 2: Attributes

Student id, Student name, mail id, mobile no.Student id, student name, mail id, mobile no.Feel free to add more columns the above one is just an example schema. Assume you are working in an Ed-tech company namely Great Learning where you will be inserting and modifying the details of the students in the Student details table. Every time the students changed their details like mobile number, You need to update theirdetails in the student details table. Here is one thing you should ensure whenever the new students' details come , you should also store them in the Student backup table so that if you modify the details in the student details table, you will be having theold details safely.You need not insert the records separately into both tables rather 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.

TABLES USED –

Student_details,Student_details_backup

QUERY –

```
CREATE TABLE Student_details
( student_id INT PRIMARY KEY,
  student_name VARCHAR(20) NOT NULL,
  mail_id VARCHAR(20) NOT NULL,
  mobile_no VARCHAR(10) NOT NULL );

CREATE TABLE Student_details_backup
( student_id INT PRIMARY KEY,
  student_name VARCHAR(20) NOT NULL,
  mail_id VARCHAR(20) NOT NULL,
  mobile_no VARCHAR(10) NOT NULL );

DELIMITER //
CREATE TRIGGER Backup_student_details
AFTER INSERT ON Student_details
FOR EACH ROW
BEGIN
Insert into Student_details_backup
values (New.student_id,New.student_name,New.mail_id,New.mobile_no);
  END //
DELIMITER ;
```

OUTCOME-

Action Output

#	Time	Action	Message
8	18:32:17	select * from Student_details LIMIT 0, 100	3 row(s) returned
9	18:32:21	select * from Student_details_backup LIMIT 0, 100	3 row(s) returned
10	18:32:37	CREATE TRIGGER Backup_student_details AFTER INSERT ON Student_details FOR EACH ROW ...	0 row(s) affected
11	18:33:25	insert into Student_details values(4,'smith','smith@4','86666')	1 row(s) affected
12	18:33:31	select * from Student_details LIMIT 0, 100	4 row(s) returned
13	18:33:36	select * from Student_details_backup LIMIT 0, 100	4 row(s) returned

CONCLUSION –

Above outcome shows that when student_details table is inserted with new value,Student_details_backup table also gets inserted with the same value.

Thank You