Common Concepts Coding Practice

In this practice set, you will get thorough with using SQL expressions, functions, case statements and set operations, that can be used perform much finer analysis.

The database consists of a *player* table that stores the information of name, match date, team, score, number of runs, number of fours and sixes scored.

Let's get started!

1. Calculate overall number of boundaries scored by each batsmen in the tournament

number of boundaries = fours + sixes

Note: Output must contain rows in the descending order of `number_of_boundaries` and `name`

Expected Output Format:

name	number_of_boundaries
...	...

SELECT name, sum(fours + sixes) AS number_of_boundaries FROM player GROUP BY name ORDER BY number_of_boundaries DESC, name DESC;

2. Get the highest score of each player who played in the year 2006

Note: Output must contain rows in the descending order of `highest_score`

Expected Output Format:

name	highest_score
...	...

SELECT name, max(score) AS highest_score FROM player WHERE CAST(strftime('%Y', match_date) AS INTEGER) = 2006 GROUP BY name ORDER BY highest_score DESC;

3. Calculate the strike rate of all the players in every match.

strike_rate = (score of the player / no_of_balls)*100

Note: Strike rate in the output should be a float value.

In SQL, when an integer is divided by another integer, it results in another integer value, i.e, 3/2 = 1 instead of a float. So we need to convert the numerator or denominator to float to get more accurate resutls, i.e 3.0/2 or 3/2.0, to get 1.5 as output.

Let's apply the same while calculating the strike_rate to get accurate results.

Note: Output must contain rows in the descending order of strike_rate

Expected Output Format:

name	match	strike_rate
...	...	...

SELECT name, MATCH, CAST(score AS float) / no_of_balls * 100 AS strike_rate FROM player ORDER BY strike_rate DESC;

4. Let's generate a performance report for all the players who played in the year 2006.

Apply the below logic to grade the player's performance.

total score	performance report
>= 150	GOOD
100<= <150	AVERAGE
<100	BELOW AVERAGE

in the year 2006

Note: Output must be in the descending order of total_score

Expected Output Format:

name	total_score	badge
...	...	...

SELECT name, sum(score) AS total_score,

CASE WHEN sum(score) >= 150 THEN "GOOD" WHEN sum(score) >= 100 AND sum(score) < 150 THEN "AVERAGE" ELSE "BELOW AVERAGE" END AS badge

FROM player WHERE CAST(strftime('%Y', match_date) AS INTEGER) = 2006

GROUP BY name ORDER BY total_score DESC;

5. For each player, get the number of matches in which their strike rate is less than 80.0, and the number of matches with strike rate greater than or equal to 80.0.

Note: Output must be in the ascending order of `name`

Expected Output Format:

name	strike_rate_less_than_80	strike_rate_greater_than_or_equal_to_80
...	...	...

SELECT NAME,

COUNT(CASE WHEN (CAST(SCORE AS FLOAT) / NO_OF_BALLS) * 100 < 80.0 THEN 1 ELSE NULL END) AS strike_rate_less_than_80,

COUNT(CASE WHEN (CAST(SCORE AS FLOAT) / NO_OF_BALLS) * 100 >= 80.0 THEN 1 ELSE NULL END) AS strike_rate_greater_than_or_equal_to_80

FROM player

GROUP BY NAME

ORDER BY name;

6. Get all the player/s who played for both *CSK* and *RCB*.

Note: Output must be in the ascending order of `name`

Expected Output Format:

name
...

SELECT name FROM player

WHERE played_for_team = 'CSK'

INTERSECT

SELECT name FROM player

WHERE played_for_team = 'RCB';

7. Get all the player/s who played only for *SRH*

Note:

- Names in the output must be in capital letters.
- Output must be in the ascending order of `name`

Expected Output Format:

name
...

SELECT upper(name) AS name FROM player

WHERE played_for_team = 'SRH'

EXCEPT

SELECT upper(name) AS name FROM player

WHERE played_for_team <> 'SRH'

ORDER BY name ASC

8. Get all the player/s who played either for *SRH*, *CSK*, or *MI*.

Note:

- Get unique players.
- Output must be in the ascending order of `name`

Expected Output Format:

name
...

SELECT name FROM player

WHERE played_for_team IN ('SRH', 'CSK')

UNION

SELECT name FROM player

WHERE played_for_team = 'MI'

ORDER BY name ASC;

9. Fetch the name, highest and lowest scores of player/s for the matches in which strike rate is greater than 50.0.

Expected Output Format:

name	highest_score	lowest_score
...	...	...

SELECT name, max(score) AS highest_score, min(score) AS lowest_score

FROM player

WHERE (CAST(score AS float) / no_of_balls) * 100 > 50.0

GROUP BY name

Player Table

id	name	score	match_date	played_for_team	match	fours	sixes	no_of_balls
1	Ravi	80	2000-05-26	CSK	CSK vs RCB	4	5	60
2	Sai	45	2006-05-30	RR	RR vs SRH	3	2	40
3	Raghav	36	2001-06-21	SRH	SRH vs MI	1	2	25
4	Ravi	72	2005-07-12	RCB	RCB vs SRH	3	4	70
5	Sai	50	2006-03-23	MI	MI vs RCB	2	3	44
6	Jadhav	50	2006-03-26	CSK	CSK vs MI	3	2	40
7	Jadhav	54	2006-05-13	CSK	CSK vs SRH	2	4	40
8	Manoj	68	2006-05-23	MI	MI vs RCB	3	3	60
9	Ravi	92	2006-05-14	SRH	SRH vs MI	4	4	90
10	Karthik	32	2007-02-13	MI	MI vs RR	1	2	28
11	Madhu	40	2002-04-13	MI	MI vs RCB	1	0	55
12	Sanjay	45	2002-04-12	SRH	SRH vs RR	2	0	60
13	Ravi	35	2002-04-13	MI	MI vs RR	0	0	50

id	name	score	match_date	played_for_team	match	fours	sixes	no_of_balls
14	Manoj	45	2005-07-15	RR	RR vs CSK	3	1	72
15	Vijay	92	2003-06-20	RR	RR vs SRH	4	3	92
16	Sai	80	2006-04-22	SRH	SRH vs RCB	2	4	80