Aggregations and Group By Coding Practice

The database consists of `player_match_details` table that stores the information of players' details like name, match, score, year, number of fours and sixes scored.

This practice set helps you get well versed with GROUP BY and HAVING clauses. Let's dive in!

1. Get the total score of each player

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

Expected Output Format:

name	total_score
...	...

SELECT name, sum(score) AS total_score FROM player_match_details GROUP BY name ORDER BY total_score DESC;

2. Get the number of half centuries scored by each player.

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

Expected Output Format:

name	half_centuries
...	...

SELECT name, count(*) AS half_centuries FROM player_match_details WHERE score >= 50 GROUP BY name ORDER BY half_centuries DESC;

3. Get the total number of 4's hit by each player as `no_of_fours` .

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

Expected Output Format:

name	no_of_fours
...	...

SELECT name, sum(fours) AS no_of_fours FROM player_match_details GROUP BY name ORDER BY no_of_fours DESC;

4. Get the highest score of every player as `max_score` .

Note: Output must contain rows in the descending order of `max_score` of the player.

Expected Output Format:

name	max_score
...	...

SELECT name, max(score) AS max_score FROM player_match_details GROUP BY name ORDER BY max_score DESC;

5. Get player `name` and the total number of matches played as `no_of_matches` by each player in the year 2012.

Note: Output must contain rows in the descending order of `no_of_matches` of each player.

Expected Output Format:

name	no_of_matches
...	...

SELECT name, count(*) AS no_of_matches FROM player_match_details WHERE year = 2012 GROUP BY name ORDER BY no_of_matches DESC;

6. Get the year-wise performance, i.e., `no_of_matches` and `runs_scored` by each player.

Note: Output must contain rows in the ascending order of `name` & `year`

Expected Output Format:

name	year	no_of_matches	runs_scored
...	...	...	...

SELECT name, year, count(*) AS no_of_matches, sum(score) AS runs_scored FROM player_match_details GROUP BY name, year ORDER BY name ASC, year ASC;

7. Get `name` , average score of players as `avg_score` , and total number of sixes scored by the players as `total_sixes` whose average score is greater than 50.

Note: Output must contain rows in the ascending order of `name` of the player.

Expected Output Format:

name	avg_score	total_sixes
...	...	...

SELECT name, avg(score) AS avg_score, sum(sixes) AS total_sixes FROM player_match_details GROUP BY name HAVING avg_score > 50 ORDER BY name ASC;

8. For each player who scored more than 50 in at least 2 matches, get the total number of matches where the players scored more than 50.

Note: Output must contain rows in the ascending order of `name` of the player.

Expected Output Format:

name	no_of_matches
...	...

SELECT name, count(*) AS no_of_matches FROM player_match_details WHERE score > 50 GROUP BY name HAVING no_of_matches >= 2 ORDER BY name ASC;

Player_match_Details Table

name	match	score	fours	sixes	year
Ram	RR vs SRH	62	2	7	2011
Joseph	SRH vs CSK	44	2	4	2012
Lokesh	DC vs DD	99	2	13	2013
David	SRH vs CSK	96	1	13	2014
Joseph	SRH vs CSK				2012

name	match	score	fours	sixes	year
Viraj	RCB vs RR	53	2	5	2010
Shyam	MI vs RCB	75	2	9	2011
Stark	MI vs DC	75	2	9	2012
Stark	MI vs SRH				2012
Ram	RR vs MI	84	1	11	2013
Joseph	SRH vs RR	42	1	4	2014
David	SRH vs MI				2014
Ramesh	CSK vs RR	9	0	0	2010
Ram	RR vs DC	75	2	9	2011
Joseph	SRH vs MI	30	5	0	2012
Lokesh	DC vs RR	87	2	11	2013
Ram	RR vs SRH				2011
David	SRH vs MI	9	0	0	2014