rye_raptor ★ 13 Last Edit: September 19, 2019 3:21 PM 1.1K VIEWS

13

**Solution 1:**
This solution works on most databases (i.e. Oracle, PostgreSQL, MySQL etc.) and on the leetcode versions.

First, we can retrieve a set of all the players along with their scores per match using "union all" i.e. first player and their scores (first score) along with second players and their scores (second score):

```
select first_player as player, first_score as score from Matches
union all
select second_player, second_score from Matches
```

From this set, we can then add the scores for each player in all matches:

```
select player, sum(score) as score from
    (select first_player as player, first_score as score from Matches
    union all
    select second_player, second_score from Matches) s
group by player
```

We can refer to this as the players' scores. To get the group of each player we can simple join this "players' scores" set to Players set based on the player_id.

However, what we really need is the maximum score in each group, therefore we can use an aggregate function here along with the join:

```
select group_id, max(score)
from Players,
    (select player, sum(score) as score from
        (select first_player as player, first_score as score from Matches
        union all
        select second_player, second_score from Matches) s
    group by player) PlayerScores
where Players.player_id = PlayerScores.player
group by group_id
```

Now that we have the maximum score in each group, it is simply a matter of retrieving all players in the "players' score" set with that score

```
select group_id, player_id
from Players,
    (select player, sum(score) as score from
        (select first_player as player, first_score as score from Matches
        union all
        select second_player, second_score from Matches) s
    group by player) PlayerScores
    where Players.player_id = PlayerScores.player
        and (group_id, score) in                    # check against maximum score in each group
            (select group_id, max(score)
            from Players,
                (select player, sum(score) as score from
                    (select first_player as player, first_score as score from Matches
                    union all
                    select second_player, second_score from Matches) s
                group by player) PlayerScores
            where Players.player_id = PlayerScores.player
            group by group_id)
```

**Final code for Solution 1:**
Finally, we take care of players with the same score by retrieving the player with the id:

```
select group_id as GROUP_ID, min(player_id) as PLAYER_ID
from Players,
    (select player, sum(score) as score from
        (select first_player as player, first_score as score from Matches
        union all
        select second_player, second_score from Matches) s
    group by player) PlayerScores
where Players.player_id = PlayerScores.player and (group_id, score) in
    (select group_id, max(score)
    from Players,
        (select player, sum(score) as score from
            (select first_player as player, first_score as score from Matches
            union all
            select second_player, second_score from Matches) s
        group by player) PlayerScores
```

**Solution 2:**
In Solution 1, the PlayerScores set had to be created twice (first to get the scores and a second time to compare the scores with the maximum score in each group). One way to remove the repeating code segment would be to use Common Table Expression (however, I do not think this can be done in leetcode). Another way might be to use variables. However, we can also leverage a trick based on how MySQL handles aggregates when **sql_mode=only_full_group_by** is disabled (which is the case in leetcode). MySQL simple picks the first item in a row when a column is in the **SELECT** clause but not the **GROUP BY** clause. Other databases would throw an error. Therefore we can simply get the players' scores (ps) like we did in Solution 1 along with the group_id of each player by joining the subquery with the Players table. Then order them to make sure the first item of each group is what we are looking for:

```
select p.group_id, ps.player_id, sum(ps.score) as score from Players p,
    (select first_player as player_id, first_score as score from Matches
    union all
    select second_player, second_score from Matches) ps
where p.player_id = ps.player_id
group by ps.player_id order by group_id, score desc, player_id
```

**Final code for Solution 2:**
Aggregating this over the group_id will then give us our final result:

```
select group_id, player_id from (
    select p.group_id, ps.player_id, sum(ps.score) as score from Players p,
        (select first_player as player_id, first_score as score from Matches
        union all
        select second_player, second_score from Matches) ps
    where p.player_id = ps.player_id
    group by ps.player_id order by group_id, score desc, player_id) top_scores
group by group_id
```

Comments: 4 Best | Most Votes | Newest to Oldest | Oldest to Newest

Type comment here... (Markdown is supported)

Post

DennyZhang ★ 225 Last Edit: September 18, 2019 1:07 PM

Not sure what mysql version leetcode supports.

If it's latest enough, functionality like this would be available: `row_number () over (partition by`

3 Reply

ufohuang ★ 4 December 21, 2019 5:43 AM

Hi, I am having serious problem and do not know why my code does not work. The output keep saying for group 3, player 20 has the highest score but the answer is clearly off. Your help is much appreciated. (I am stuck on this and spent half an hour trying to figure it out why QAQ)

SELECT GROUP_ID, P.PLAYER_ID
FROM players p
JOIN
(SELECT player_id, SUM(score) AS score
FROM
(SELECT m1.first_player AS player_id, SUM(first_score) AS score
FROM matches m1
GROUP BY first_player

Read More

0 Reply

pdeepthy ★ 4 September 26, 2019 3:37 AM

Very good explanation. Would like to see more from you !!!

0 Reply

lillianru ★ 8 September 19, 2019 12:52 AM

I think mine logic works the same as yours but couldnt go through...

with temp as(
select group_id,player_id,sum(score) t_score
from(select first_player player_id,first_score score
from matches
union
select second_player player_id,second_score score
from matches) t1
left join Players t2
on t1.player_id = t2.player_id

Read More

0 Show 1 reply Reply