Decode Gaming Behaviour SQL problem statement Analysis

```
Create table Player_details
(
        P_id int8 primary key,
        Pname varchar(50),
        L1_status int8,
        L2_status int8,
        L1_code varchar(50),
        L2_code varchar(50)
)
select * from Player_details
Create table Level_details
(
        P_id int8 ,
        Dev_ID varchar(50),
        Time_stamp timestamp,
        Stages_crossed int8,
        Level int8,
        Difficulty varchar(15),
        Kill_Count int8,
        Headshots_Count int8,
        Score int8,
        Lives_Earned int8,
)
select * from Level_details


-- 1. Extract `P_ID`, `Dev_ID`, `PName`, and `Difficulty_level` of all players at Level 0.


select p.p_id , l.Dev_id,p.Pname,l.difficulty
from player_details as p join level_details as l
on p.p_id = l.p_id
where l.levels=0
order by p_id
```

Decode Gaming Behaviour SQL problem statement Analysis

-- 2) Find `Level1_code`wise average `Kill_Count` where `lives_earned` is 2, and at least 3 stages are crossed.

```
Select pd.l1_code , avg(ld.kill_count) as avg_killcount
from player_details as pd  join level_details as ld
on pd.p_id = ld.p_id
where  ld.lives_earned=2 and ld.stages_crossed>=3
group by pd.l1_code;
```

-- 3. Find the total number of stages crossed at each difficulty level for Level 2 with players
-- using `zm_series` devices. Arrange the result in decreasing order of the total number of
-- stages crossed.

```
Select difficulty , sum(stages_crossed) as total_stagescrossed
from level_details
where levels=2 and dev_id like 'zm%'
group by difficulty
order by  sum(stages_crossed) desc;
```

-- 4. Extract `P_ID` and the total number of unique dates for those players who have played
-- games on multiple days.

```
select P_id , Count(Distinct(date(time_stamp)))as total_days
from level_details
group by P_id
having Count(Distinct(date(time_stamp))) >1
order by Count(Distinct(date(time_stamp))) desc
```

5-- Find `P_ID` and levelwise sum of `kill_counts` where `kill_count` is greater than the
-- average kill count for Medium difficulty.

```
select P_id,levels,sum(Kill_count) as kill_counts
```

Decode Gaming Behaviour SQL problem statement Analysis

```
from level_details
where kill_count>(select avg(kill_count) from level_details where difficulty like 'Medium')
group by p_id ,levels
order by p_id ,levels

--6. Find `Level` and its corresponding `Level_code`wise sum of lives earned, excluding Level
-- 0. Arrange in ascending order of level.

select ld.levels , pd.l1_code , sum(lives_earned) as total_livesearned
from level_details as ld join player_details as pd
on ld.P_id = pd.P_id
where ld.levels>0
group by ld.levels , pd.l1_code
order by ld.levels desc

-- 7. Find the top 3 scores based on each `Dev_ID` and rank them in increasing order using
-- `Row_Number`. Display the difficulty as well.

Select * from (Select score , dev_id , difficulty,
row_number() over(partition by dev_id order by score desc) as ranks
from level_details) as results
where ranks<=3

-- 8. Find the `first_login` datetime for each device ID.

select * from ( select dev_id , time_stamp ,
row_number()over(partition by dev_id order by time_stamp) as rankss
from level_details) as result
where rankss=1

-- 9. Find the top 5 scores based on each difficulty level and rank them in increasing order
-- using `Rank`. Display `Dev_ID` as well.
```

Decode Gaming Behaviour SQL problem statement Analysis

```
Select * from (Select dev_id , score , difficulty,
rank() over(partition by difficulty order by score desc) as ranks
from level_details) as results
where ranks<=5

-- 10. Find the device ID that is first logged in (based on `start_datetime`) for each player
-- (`P_ID`). Output should contain player ID, device ID, and first login datetime.



select * from ( select p_id , dev_id , time_stamp ,
                        rank()over(partition by p_id order by time_stamp) as rankss
                        from level_details) as result
                        where rankss=1

-- 11. For each player and date, determine how many `kill_counts` were played by the player
-- so far.
-- a) Using window functions
-- b) Without window functions

-- b) Without window functions
Select p_id,Date(time_stamp) as Dates,Sum(kill_count) as Kill_Counts
From level_details
Group by p_id,Date(time_stamp)
order by p_id,Date(time_stamp)

-- a) Using window functions

select p_id,dates ,player_killcounts from
(select p_id , date(time_stamp) as dates, sum(kill_count) over(partition by p_id ,date(time_stamp)) as
player_killcounts,
        row_number() over(partition by p_id ,date(time_stamp)) as row_no
        from level_details ) as result
        where row_no =1
```

Decode Gaming Behaviour SQL problem statement Analysis

-- 12. Find the cumulative sum of stages crossed over `start_datetime` for each `P_ID`,

-- excluding the most recent `start_datetime`.

```
Select P_ID,time_stamp, stages_crossed,Cumulative_of_Stages_crossed
from ( Select P_ID,time_stamp,stages_crossed,
sum(stages_crossed) Over(partition by P_ID Order by
time_stamp) as Cumulative_of_Stages_crossed,
Row_Number() Over(Partition by P_ID) as Row_NO
From level_details
) as result
Where (P_ID,Row_NO)
NOT IN
(Select P_ID,MAX(Row_No) as Row_No
From (Select P_ID,time_stamp,
sum(stages_crossed) Over(partition by P_ID Order
by time_stamp) as Cumulative_of_Stages_crossed,
Row_Number() Over(Partition by P_ID) as Row_NO
From level_details) as Project
Group by P_ID);
```

-- 13. Extract the top 3 highest sums of scores for each `Dev_ID` and the corresponding `P_ID`.

```
select p_id, dev_id , sum_ofscore from
(select p_id , dev_id ,sum_ofscore,rank() over(partition by dev_id order by sum_ofscore desc ) as
row_no from
(select p_id,dev_id,sum(score) as sum_ofscore from level_details
group by p_id,dev_id
order by sum_ofscore desc,dev_id asc ) as result1) as result2
where row_no<=3
```

-- 14. Find players who scored more than 50% of the average score, scored by the sum of

-- scores for each `P_ID`.

Decode Gaming Behaviour SQL problem statement Analysis

```
Select * from ( select p_id , sum(score) as sumofscore from level_details group by p_id order by p_id asc)

where sumofscore>( select 0.5 * avg(sumofscore) as avg_score from (select p_id,sum(score) as sumofscore

from level_details group by p_id ))


-- 15. Create a stored procedure to find the top `n` `headshots_count` based on each `Dev_ID` and rank them in increasing order using `Row_Number`. Display the difficulty as well.


Delimiter//

Create procedure Top_N(IN Num int)

begin

select dev_id,headshots_count,difficulty from (select dev_id ,difficulty,headshots_count,

row_number()over (partition by dev_id order by headshots_count)as row_no

from level_details) as result

where row_no<=Num;

end//


call Top_N(3);
```