Decode Gaming Behavior

1) Extract `P\_ID`, `Dev\_ID`, `PName`, and `Difficulty\_level` of all players at Level 0.

Select T1.P\_ID,

T2.Dev\_ID,

T1.PName,

T2.Difficulty as Difficulty\_Level

From player\_details as T1 Inner Join level\_details as T2 Using(P\_ID)

Where T2.Level=0;

2) Find `Level1\_code`wise average `Kill\_Count` where `lives\_earned` is 2, and at least 3 stages are crossed.

Select T1.L1\_Code,

avg(T2.Kill\_Count) as Avg\_Kill\_Count

From player\_details as T1 Inner Join level\_details as T2 Using(P\_ID)

Where T2.Stages\_crossed>=3 and T2.Lives\_Earned=2

Group by T1.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 as Difficulty\_Level,Sum(stages\_crossed) as "No. of Stages Crossed"

From level\_details

where level=2 and Dev\_ID like "zm%"

Group by Difficulty\_Level

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 T1.P\_ID, Count(Distinct(Date(T2.start\_datetime))) as "No.of Unique Dates"

From player\_details as T1 Inner Join level\_details as T2 Using(P\_ID)

Group by T1.P\_ID

Having Count(Distinct(Date(T2.start\_datetime)))>1;

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,Level,Sum(kill\_count) as Kill\_Count

From level\_details

Where kill\_count >(Select avg(kill\_count) From level\_details where Difficulty="Medium")

Group by P\_ID,Level

order by P\_ID,Level;

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

Select T2.Level,T1.L1\_Code as Level\_Code,Sum(T2.Lives\_Earned) as Lives\_Earned

From player\_details as T1 Inner Join level\_details as T2 Using(P\_ID)

Where T2.Level<>0

Group by T2.Level,T1.L1\_Code

Union All

Select T2.Level,T1.L2\_Code as Level\_Code,Sum(T2.Lives\_Earned) as Lives\_Earned

From player\_details as T1 Inner Join level\_details as T2 Using(P\_ID)

Where T2.Level<>0

Group by T2.Level,T1.L2\_Code

Order by Level;

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 Dev\_ID,Difficulty,Score,Row\_Number() Over(Partition by Dev\_ID Order by Score desc) as Rankk From Level\_Details) as Project

Where Rankk<=3;

8) Find the `first\_login` datetime for each device ID.

Select Dev\_ID,Start\_datetime

From (Select Dev\_ID,start\_datetime,Row\_number() Over(Partition by Dev\_ID Order by start\_datetime) as Rankk

From level\_details) as Project

Where Rankk=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.

Select \*

From (Select Dev\_ID,Difficulty,Score,Rank() Over(Partition by Difficulty Order by Score desc) as Rankk From Level\_details) as Project

Where Rankk<=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 P\_ID,Dev\_ID,Start\_Datetime

From (Select P\_ID,Dev\_ID,Start\_Datetime,Rank() Over(Partition by P\_ID Order by start\_Datetime) as Rankk From Level\_details) as Project

Where Rankk=1;

11) For each player and date, determine how many `kill\_counts` were played by the player so far.

a) Using window functions

Select P\_ID,Datee,Kill\_Count

From (Select P\_ID,Date(Start\_datetime) as Datee,SUM(kill\_Count) Over(Partition by P\_ID,Date(Start\_datetime)) as Kill\_Count,

Row\_Number() Over(Partition by P\_ID,Date(Start\_datetime)) as Row\_No

From level\_details) as Project

Where Row\_No=1;

b) Without window functions

Select P\_ID,Date(Start\_datetime) as Datee,Sum(kill\_count) as Kill\_Count

From level\_details

Group by P\_ID,Date(Start\_datetime);

12) Find the cumulative sum of stages crossed over a start\_datetime for each `P\_ID`

Select P\_ID,start\_datetime, sum(stages\_crossed) Over(partition by P\_ID Order by Start\_datetime) as Cumulative\_of\_Stages\_crossed

From level\_details;

13) Find the cumulative sum of stages crossed over `start\_datetime` for each `P\_ID`, excluding the most recent `start\_datetime`.

Select P\_ID,Start\_datetime,Cumulative\_of\_Stages\_crossed

From ( Select P\_ID,start\_datetime,

sum(stages\_crossed) Over(partition by P\_ID Order by

Start\_datetime) as Cumulative\_of\_Stages\_crossed,

Row\_Number() Over(Partition by P\_ID) as Row\_NO

From level\_details

) as Mentorness

Where (P\_ID,Row\_NO)

NOT IN

(Select P\_ID,MAX(Row\_No) as Row\_No

From (Select P\_ID,start\_datetime,

sum(stages\_crossed) Over(partition by P\_ID Order

by Start\_datetime) as Cumulative\_of\_Stages\_crossed,

Row\_Number() Over(Partition by P\_ID) as Row\_NO

From level\_details) as Project

Group by P\_ID);

14) Extract the top 3 highest sums of scores for each `Dev\_ID` and the corresponding `P\_ID`.

Select Dev\_ID,P\_ID,Total\_Score

From (Select Dev\_ID,P\_ID,Total\_Score,Row\_Number() Over(Partition by Dev\_ID order by Total\_Score desc) as Rankk

From(

Select Dev\_ID,P\_ID,Sum(Score) as Total\_Score

From level\_details

Group by Dev\_ID,P\_ID

Order by Dev\_ID asc,Total\_Score desc

) as Project) as Mentorness

Where Rankk<=3;

15) Find players who scored more than 50% of the average score, scored by the sum of scores for each `P\_ID`

Select \*

From (Select P\_ID, Sum(Score) as Total\_Score From level\_details Group by P\_ID) as Mentorness

Where Total\_Score>(Select 0.5\*Avg(Total\_Score) As Avg\_Score From ( Select P\_ID, Sum(Score) as Total\_Score From level\_details Group by P\_ID) as Project);

16) 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 TopN(IN P\_TopN Int)

Begin

Select Dev\_ID,Difficulty,Headshots\_Count

From

(

Select Dev\_ID,Difficulty,Headshots\_Count,Row\_Number() Over(Partition by Dev\_ID Order by Headshots\_Count) as Rankk

From level\_details

) as Project

Where Rankk<=P\_TopN;

End //

Call TopN(3);

17) Create a function to return sum of Score for a given player\_id.

Delimiter //

Create Function Get\_Score(F\_ID Int)

Returns Int

Deterministic

Begin

Return

(Select Sum(Score) From level\_details Where P\_ID=F\_ID);

End //

Select Get\_Score(300) as score;