Game Analysis

WITH SQL

The project goal is to analyze gaming behavior

- I will be working with a dataset related to a game. The dataset includes two tables:

Player_Details

Level_Details2

Dataset Description

Player Details Table 1:

- P_ID`: Player ID
- P_Name`: Player Name
- L1_status`: Level 1 Status
- L2_status`: Level 2 Status
- L1_code`: System generated Level 1 Code
- L2_code`: System generated Level 2 Code

Level Details Table 2:

- P_ID`: Player ID
- Dev_ID`: Device ID
- Start_time: Start Time
- Stages_crossed: Stages Crossed
- level`: Game Level
- difficulty: Difficulty Level
- Kill_coun: Kill Count
- Head shots_count: Headshots Count
- score`: Player Score
- Lives_earned: Extra Lives Earned

Query 1

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

- Select PD.P_ID , LD.Dev_ID , PD.PName , LD.Difficulty AS Difficulty_level from player_details AS PD JOIN level_details2 AS LD ON PD.P_ID = LD.P_ID WHERE LD.level = 0 ;

The Output

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

P_ID	Dev_ID	PName	Difficulty_level
656	rf_013	sloppy-denim-wolfhound	Medium
358	zm_013	skinny-grey-quetzal	Medium
358	zm_017	skinny-grey-quetzal	Low
632	bd_013	dorky-heliotrope-barracuda	Difficult
429	bd_013	flabby-firebrick-bee	Medium
310	bd_015	gloppy-tomato-wasp	Difficult
211	bd_017	breezy-indigo-starfish	Low
300	zm_015	lanky-asparagus-gar	Difficult
641	rf_013	homey-alizarin-gar	Difficult
641	rf_015	homey-alizarin-gar	Medium
641	rf_013	homey-alizarin-gar	Low
558	wd_019	woozy-crimson-hound	Difficult

Result Grid | Form Editor | Field Types

Result 2 | Read Only

Query 2

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

- select pd.L1_code as Level1_code , avg(ld.Kill_Count) as wiseaverage from player_details pd JOIN level_details2 ld ON pd.P_ID = ld.P_ID
  where ld.lives_earned = 2 and ld.stages_crossed >= 3 GROUP BY pd.L1_code;

The Output

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Level1_code	wiseaverage
speed_blitz	19.3333
war_zone	19.2857
bulls_eye	22.2500

Result Grid

Form Editor

Result 3

Read Only

Query 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 sum(ld.stages_crossed) as totalnumber, LD.Difficulty AS Difficulty_level from player_details pd JOIN level_details2 ld ON pd.P_ID = ld.P_ID
  where LD.level = 2 and LD.Dev_ID REGEXP'^zm' group by LD.Difficulty order by totalnumber DESC;

The Output

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

totalnumber	Difficulty_level
46	Difficult
35	Medium
15	Low

Result Grid

Form Editor

Result 4

Read Only

Output

Query 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(TimeStamp)) AS Total_Unique_Dates FROM level_details2 GROUP BY P_ID HAVING COUNT(DISTINCT DATE(TimeStamp)) > 1;

The Output

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

P_ID	Total_Unique_Dates
211	4
224	2
242	2
292	2
300	3
310	3
368	2
483	3
590	3
632	3
641	2
644	2
656	4
683	4

Result Grid | Form Editor | Field Types | Query Stats

Result 5 | Read Only

Query 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 total_Kill_Count FROM level_details2 WHERE Kill_Count > (SELECT AVG(Kill_Count) FROM level_details2 WHERE Difficulty = 'Medium') GROUP BY P_ID, Level;

The Output

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

P_ID	Level	total_Kill_Count
644	2	24
656	1	37
632	0	45
632	1	28
632	2	53
429	1	30
429	2	55
310	1	20
310	0	34
211	0	20
211	1	55
547	1	20
300	1	48
224	1	54
224	2	58
242	1	58
292	1	21

Result 6

Output

Result Grid | Form Editor | Field Types | Query Stats | Read Only

Query 6

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

- SELECT ld.level AS Level, pd.L2_Code AS Level_code, SUM(ld.lives_earned) AS Total_Lives_Earned FROM Level_Details2 ld JOIN Player_Details pd ON ld.P_ID = pd.P_ID WHERE ld.level != 0 GROUP BY ld.level, pd.L2_Code ORDER BY ld.level ASC;

The Output

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Level	Level_code	Total_Lives_Earned
1		7
1	cosmic_vision	5
1	resurgence	1
1	splippery_slope	10
2	cosmic_vision	12
2	resurgence	11
2	splippery_slope	28

Result Grid | Form Editor

Result 1 | Read Only

Query 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 R_ank, Dev_ID, Score ,Difficulty FROM (SELECT row_number() OVER (PARTITION BY Dev_ID ORDER BY Score ASC) AS R_ank, Dev_ID,Score, Difficulty FROM level_details2) AS RankedLevels WHERE R_ank <= 3;

The Output

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

R_ank	Dev_ID	Score	Difficulty
1	bd_013	100	Difficult
2	bd_013	100	Difficult
3	bd_013	540	Low
1	bd_015	380	Low
2	bd_015	1050	Medium
3	bd_015	1300	Difficult
1	bd_017	390	Low
2	bd_017	1750	Medium
3	bd_017	2400	Low
1	rf_013	100	Medium
2	rf_013	100	Medium
3	rf_013	105	Low
1	rf_015	40	Medium
2	rf_015	150	Low
3	rf_015	670	Medium
1	rf_017	280	Difficult
2	rf_017	1100	Difficult
3	rf_017	3500	Difficult
1	wd_019	100	Difficult

Result 2 | Read Only

Query 8

Find the `first_login` datetime for each device ID.

- select Dev_ID , min(TimeStamp) as first_login from level_details2 group by Dev_ID ;

The Output

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Dev_ID	first_login
zm_015	2022-10-11 14:05:08
rf_015	2022-10-11 19:34:25
bd_017	2022-10-12 07:30:18
rf_013	2022-10-11 05:20:40
bd_015	2022-10-11 18:45:55
rf_017	2022-10-11 09:28:56
bd_013	2022-10-11 02:23:45
zm_017	2022-10-11 14:33:27
zm_013	2022-10-11 13:00:22
wd_019	2022-10-12 23:19:17

2022-10-11 14:33:27

Result Grid | Form Editor | Field Types

Result 3 | Read Only

Query 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 R_ank, Dev_ID, Score ,Difficulty FROM (SELECT row_number() OVER (PARTITION BY Difficulty ORDER BY Score ASC) AS R_ank, Dev_ID,Score, Difficulty FROM level_details2) AS RankedLevels WHERE R_ank <= 5;

The Output

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

R_ank	Dev_ID	Score	Difficulty
1	bd_013	100	Difficult
2	zm_017	100	Difficult
3	wd_019	100	Difficult
4	bd_013	100	Difficult
5	rf_013	235	Difficult
1	zm_017	50	Low
2	zm_017	70	Low
3	rf_013	105	Low
4	rf_015	150	Low
5	bd_015	380	Low
1	rf_015	40	Medium
2	rf_013	100	Medium
3	zm_015	100	Medium
4	rf_013	100	Medium
5	zm_013	120	Medium

Result 4 | Read Only

Result Grid | Form Editor | Field Types | Query Stats

Query 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 ld.P_ID,ld.Dev_ID,ld.TimeStamp AS first_login_datetime FROM Level_Details2 ld INNER JOIN (SELECT P_ID, MIN(TimeStamp) AS min_start_time FROM Level_Details2 GROUP BY P_ID) AS sub ON ld.P_ID = sub.P_ID AND ld.TimeStamp = sub.min_start_time;

The Output

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

P_ID	Dev_ID	first_login_datetime
644	zm_015	2022-10-11 14:05:08
656	bd_013	2022-10-11 17:47:09
296	zm_017	2022-10-14 15:15:15
632	bd_013	2022-10-12 16:30:30
428	bd_015	2022-10-15 18:00:00
429	rf_017	2022-10-11 09:28:56
310	rf_017	2022-10-11 15:15:15
211	bd_017	2022-10-12 13:23:45
319	zm_017	2022-10-12 14:20:40
547	bd_013	2022-10-15 02:19:27
300	rf_013	2022-10-11 05:20:40
224	rf_017	2022-10-14 01:15:56
242	bd_013	2022-10-13 01:14:29
292	rf_013	2022-10-12 04:29:45
590	bd_017	2022-10-12 07:30:18
483	zm_017	2022-10-11 14:33:27
368	zm_015	2022-10-12 01:14:34
663	wd_019	2022-10-15 06:30:20
683	bd_013	2022-10-11 02:23:45

Result 5

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Read Only

Query 11

For each player and date, determine how many `kill_counts` were played by the player so far.

❖ Without window functions

```
SELECT PD.P_ID,PD.PName,DATE(LD.TimeStamp) AS Date,SUM(LD.kill_count) AS Total_kill
FROM  player_details AS PD JOIN level_details2 AS LD ON PD.P_ID = LD.P_ID GROUP BY PD.P_ID,
PD.PName, DATE(LD.TimeStamp);
```

The Output

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

P_ID	PName	Date	Total_kill
644	randy-turquoise-scorpion	2022-10-11	18
644	randy-turquoise-scorpion	2022-10-12	24
656	sloppy-denim-wolfhound	2022-10-15	15
656	sloppy-denim-wolfhound	2022-10-13	19
656	sloppy-denim-wolfhound	2022-10-14	3
656	sloppy-denim-wolfhound	2022-10-11	18
296	silly-taupe-ray	2022-10-14	11
632	dorky-heliotrope-barracuda	2022-10-12	73
632	dorky-heliotrope-barracuda	2022-10-13	27
632	dorky-heliotrope-barracuda	2022-10-14	30
428	leaky-magnolia-iguana	2022-10-15	5
429	flabby-firebrick-bee	2022-10-11	99
310	gloppy-tomato-wasp	2022-10-11	20
310	gloppy-tomato-wasp	2022-10-13	34
310	gloppy-tomato-wasp	2022-10-15	14
211	breezy-indigo-starfish	2022-10-12	45
211	breezy-indigo-starfish	2022-10-13	44
211	breezy-indigo-starfish	2022-10-14	9
211	breezy-indigo-starfish	2022-10-15	15

Result 8

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Read Only

Query 11

For each player and date, determine how many `kill_counts` were played by the player so far.

- Using window functions
- SELECT PD.P_ID,PD.PName, DATE(LD.TimeStamp) AS Date, SUM(SUM(LD.kill_count)) OVER (PARTITION BY PD.P_ID, DATE(LD.TimeStamp)) AS Total_kill_for_player_date FROM player_details AS PD JOIN level_details2 AS LD ON PD.P_ID = LD.P_ID GROUP BY PD.P_ID, PD.PName, DATE(LD.TimeStamp);

The Output

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

P_ID	PName	Date	Total_kill
644	randy-turquoise-scorpion	2022-10-11	18
644	randy-turquoise-scorpion	2022-10-12	24
656	sloppy-denim-wolfhound	2022-10-15	15
656	sloppy-denim-wolfhound	2022-10-13	19
656	sloppy-denim-wolfhound	2022-10-14	3
656	sloppy-denim-wolfhound	2022-10-11	18
296	silly-taupe-ray	2022-10-14	11
632	dorky-heliotrope-barracuda	2022-10-12	73
632	dorky-heliotrope-barracuda	2022-10-13	27
632	dorky-heliotrope-barracuda	2022-10-14	30
428	leaky-magnolia-iguana	2022-10-15	5
429	flabby-firebrick-bee	2022-10-11	99
310	gloppy-tomato-wasp	2022-10-11	20
310	gloppy-tomato-wasp	2022-10-13	34
310	gloppy-tomato-wasp	2022-10-15	14
211	breezy-indigo-starfish	2022-10-12	45
211	breezy-indigo-starfish	2022-10-13	44
211	breezy-indigo-starfish	2022-10-14	9
211	breezy-indigo-starfish	2022-10-15	15

Result 8

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Read Only

Query 12

Find the cumulative sum of stages crossed over `start_datetime` for each `P_ID`, excluding the most recent `start_datetime`.

- select P_ID ,DATE(TimeStamp) as date ,sum(Stages_crossed) as Total from level_details2 where DATE(TimeStamp) != (select max(DATE(TimeStamp)) from level_details2) group by P_ID, DATE(TimeStamp) ;

The Output

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

P_ID	date	Total
644	2022-10-11	4
644	2022-10-12	6
656	2022-10-13	4
656	2022-10-14	2
656	2022-10-11	10
296	2022-10-14	6
632	2022-10-12	10
632	2022-10-13	12
632	2022-10-14	8
429	2022-10-11	25
310	2022-10-11	7
310	2022-10-13	5
211	2022-10-12	9
211	2022-10-13	10
211	2022-10-14	7
319	2022-10-12	7
300	2022-10-11	12
300	2022-10-12	5
300	2022-10-13	3

Result 9

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Read Only

Output

Query 13

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

- SELECT P_ID, Dev_ID, R_ank, Total_Score FROM (SELECT P_ID, Dev_ID, ROW_NUMBER() OVER (PARTITION BY Dev_ID ORDER BY SUM(Score) DESC) AS R_ank, SUM(Score) AS Total_Score FROM level_details2 GROUP BY P_ID, Dev_ID) AS RankedScores WHERE R_ank <= 3;

The Output

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

P_ID	Dev_ID	R_ank	Total_Score
224	bd_013	1	9870
310	bd_013	2	3370
211	bd_013	3	3200
310	bd_015	1	5300
683	bd_015	2	3200
368	bd_015	3	1950
590	bd_017	1	2400
644	bd_017	2	1750
211	bd_017	3	390
368	rf_013	1	2970
211	rf_013	2	2700
300	rf_013	3	2300
483	rf_015	1	3950
683	rf_015	2	2800
590	rf_015	3	900
310	rf_017	1	5140
224	rf_017	2	5140
429	rf_017	3	3500
483	wd_019	1	4390

Result 10

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Read Only

Query 14

Find players who scored more than 50% of the average score, scored by the sum of scores for each `P_ID`.

❖ SELECT P_ID, PName , total_score FROM (SELECT pd.P_ID, pd.PName, SUM(ld.score) AS total_score, AVG(SUM(ld.score)) OVER() AS avg_total_score FROM Player_Details pd INNER JOIN level_details2 ld ON pd.P_ID = ld.P_ID GROUP BY pd.P_ID, pd.PName) AS subquery WHERE total_score > 0.5 * avg_total_score;

The Output

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

P_ID	PName	total_score
656	sloppy-denim-wolfhound	4820
632	dorky-heliotrope-barracuda	10750
429	flabby-firebrick-bee	13220
310	gloppy-tomato-wasp	1381 13810
211	breezy-indigo-starfish	1094
300	lanky-asparagus-gar	4860
224	nippy-peach-neanderthal	16310
242	slaphappy-cinnamon-squirrel	6310
590	stealthy-xanthic-cattle	8000
483	tasty-peach-fly	17230
368	homely-vermilion-toad	8710
663	fuzzy-cornflower-whippet	10750
683	craggy-ivory-dragonfly	18140

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Result 11 ×

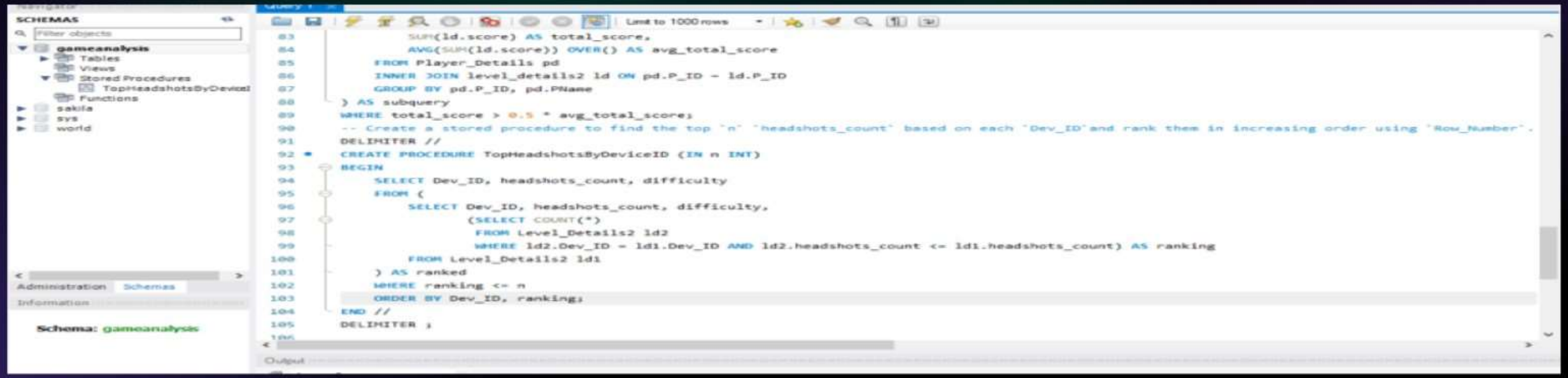
Read Only

Query 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 TopHeadshotsByDeviceID (IN n INT) BEGIN SELECT Dev_ID, headshots_count, difficulty FROM (SELECT Dev_ID, headshots_count, difficulty, (SELECT COUNT(*) FROM Level_Details2 ld2 WHERE ld2.Dev_ID = ld1.Dev_ID AND ld2.headshots_count <= ld1.headshots_count) AS ranking FROM Level_Details2 ld1) AS ranked WHERE ranking <= n ORDER BY Dev_ID, ranking; END // DELIMITER ;
- call TopHeadshotsByDeviceID (N);

The Output



Thank you