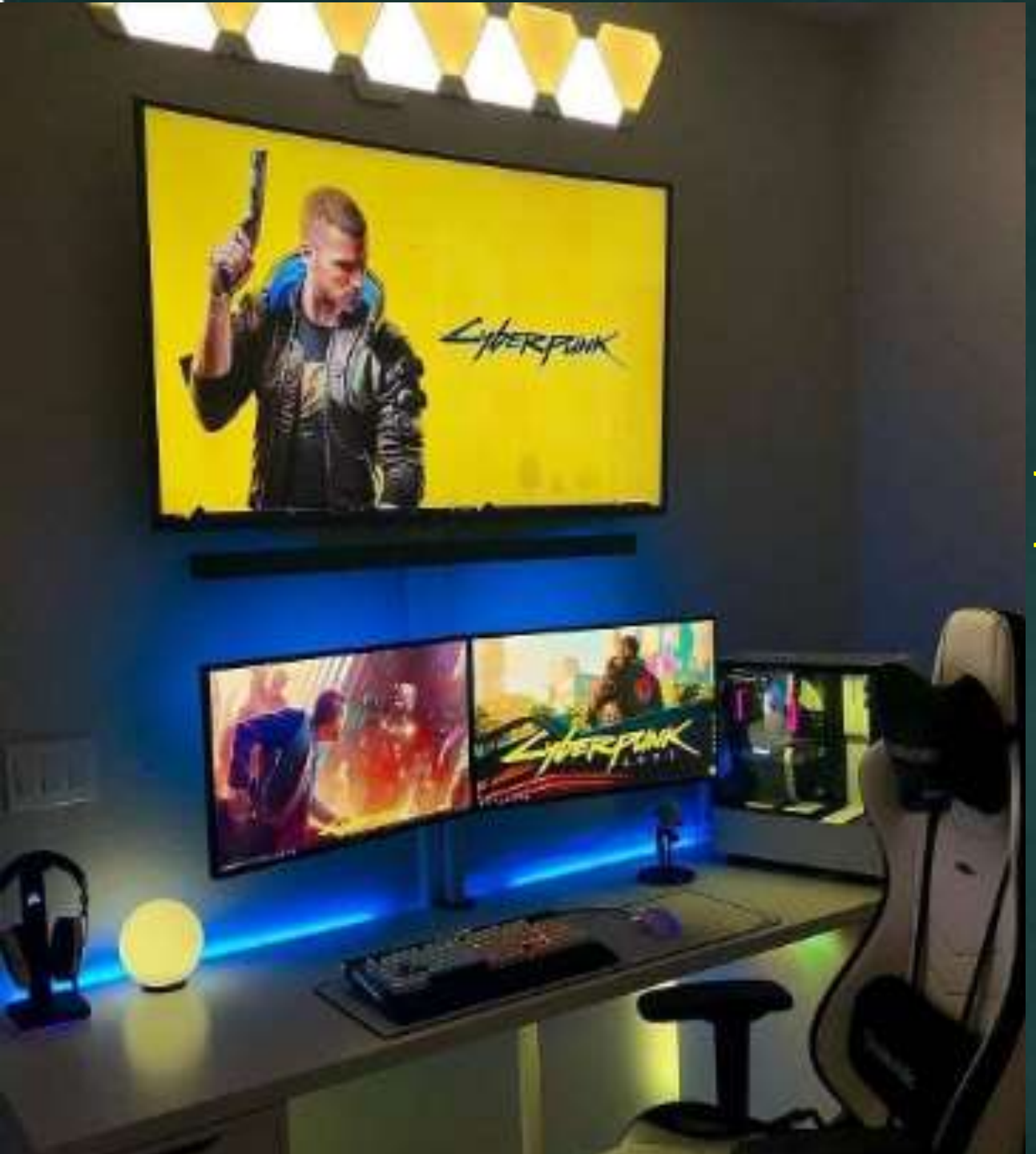
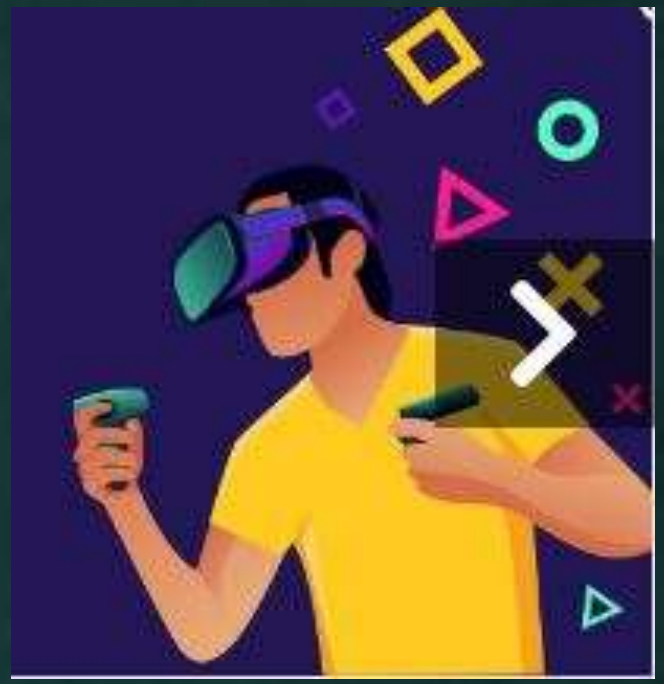


Decode Gaming Behaviour Analysis

A SQL Project Analysis by Neha Goyal

Contents



About the Case Study

Decode Gaming Behaviour project contains the records of the players. The information's are such as player id, player name, device id etc.

This project aims to analyse player behavior and performance within the game, providing valuable insights that can be used to enhance the gaming experience and optimize game design.

Objective of the Analysis

1

By Analysing the data in the "Player Details" and "Level Details" tables, you can identify patterns in player behaviour, such as how they progress through different levels, the strategies they use to overcome challenges, and the factors that contribute to their success or failure.

2

By understanding player behaviour and preferences, developers can make informed decisions about game features, content, and marketing strategies, ultimately leading to a more engaging and satisfying user experience.

3

Analyse the player's Highest Scoring , total Stages crossed by player, top Headshot counts ,levels Crossed by Player, Highest Kill counts and Login Time.

Knowing the Dataset

Table 1

The "**Player Details**" table contains key information about each player, including their unique Player ID (P_ID), Player Name (PName), and the status of their progress in Level 1 (L1_status) and Level 2 (L2_status) and their codes .

Table 2

The "**Level Details**" table provides a comprehensive view of player performance at each level of the game. It includes information such as the Player ID (P_ID), Device ID (Dev_ID), start time (start_time), stages crossed (stages_crossed), game level (level), difficulty level (difficulty), kill count (kill_count), headshots count (headshots_count), player score (score), and extra lives earned (lives_earned).

Data Structure

Player Details

P_id int8 primary key,

Pname varchar(50),

L1_status int8,

L2_status int8,

L1_code varchar(50),

L2_code varchar(50)

Level Details

P_id int8 primary key,

Dev_ID varchar(50) primary key,

Time_stamp timestamp primary key,

Stages_crossed int8,

Level int8,

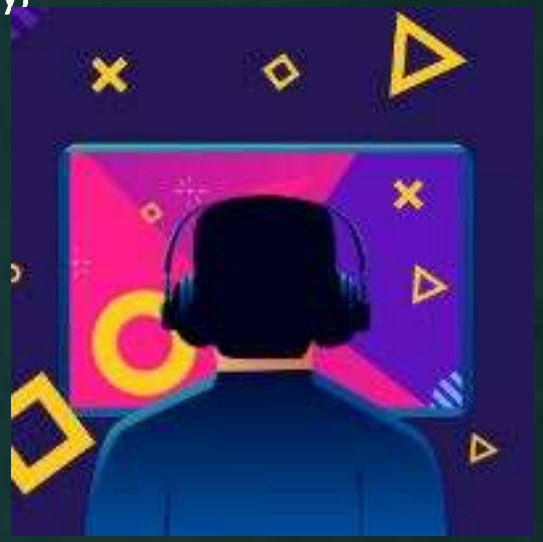
Difficulty varchar(15),

Kill_Count int8,

Headshots_Count int8,

Score int8,

Lives_Earned int8,



SQL Queries Analysis

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

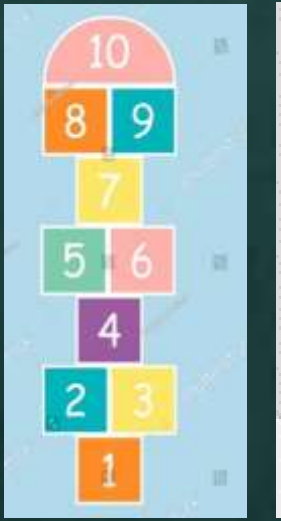
```
Select p.p_id , l.Dev_id,p.Pname,l.difficulty
from player_details as p join level_details as l
on p.p_id = l.p_id
where l.levels=0
order by p_id
```

	p_id bigint	dev_id character varying (10)	pname character varying (50)	difficulty character varying (15)
1	211	bd_017	breezy-indigo-starfish	Low
2	300	zm_015	lanky-asparagus-gar	Difficult
3	310	bd_015	gloppy-tomato-wasp	Difficult
4	358	zm_017	skinny-grey-quetzal	Low
5	358	zm_013	skinny-grey-quetzal	Medium
6	429	bd_013	flabby-firebrick-bee	Medium
7	558	wd_019	woozy-crimson-hound	Difficult
8	632	bd_013	dorky-heliotrope-barracuda	Difficult
9	641	rf_013	horney-alizarin-gar	Low
10	641	rf_015	horney-alizarin-gar	Medium
11	641	rf_013	horney-alizarin-gar	Difficult
12	656	rf_013	sloppy-denim-wolfhound	Medium

SQL Queries 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;
```



Data Output

	l1_code character varying (50)	avg_killcount numeric
1	bulls_eye	22.2500000000000000
2	war_zone	19.2857142857142857
3	speed_blitz	19.3333333333333333

SQL Queries Analysis

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;
```

Data Output

	difficulty character varying (15)	total_stagescrossed numeric
1	Difficult	46
2	Medium	35
3	Low	15

SQL Queries Analysis

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
```

Data Output

	p_id bigint	total_days bigint
1	211	4
2	656	4
3	683	4
4	632	3
5	310	3
6	300	3
7	483	3
8	590	3
9	292	2
10	242	2
11	368	2
12	641	2
13	644	2
14	224	2

SQL Queries Analysis

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

Data Output

	p_id bigint	levels bigint	kill_counts numeric
1	211	0	20
2	211	1	55
3	224	1	54
4	224	2	58
5	242	1	58
6	292	1	21
7	300	1	48
8	310	0	34
9	310	1	20
10	368	1	20
11	368	2	24
12	429	1	30
13	429	2	55
14	483	1	40
15	483	2	94
16	547	1	20
17	558	0	21
18	590	1	24
19	632	0	45
20	632	1	28

SQL Queries Analysis

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
```

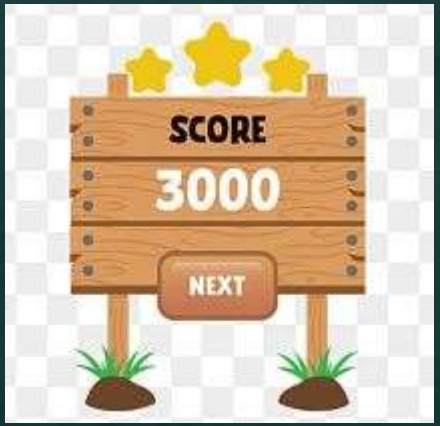
Data Output

	levels bigint	l1_code character varying (50)	total_livesearned numeric
1	2	bulls_eye	14
2	2	speed_blitz	20
3	2	war_zone	17
4	1	bulls_eye	5
5	1	leap_of_faith	0
6	1	speed_blitz	7
7	1	war_zone	11

SQL Queries Analysis

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
```



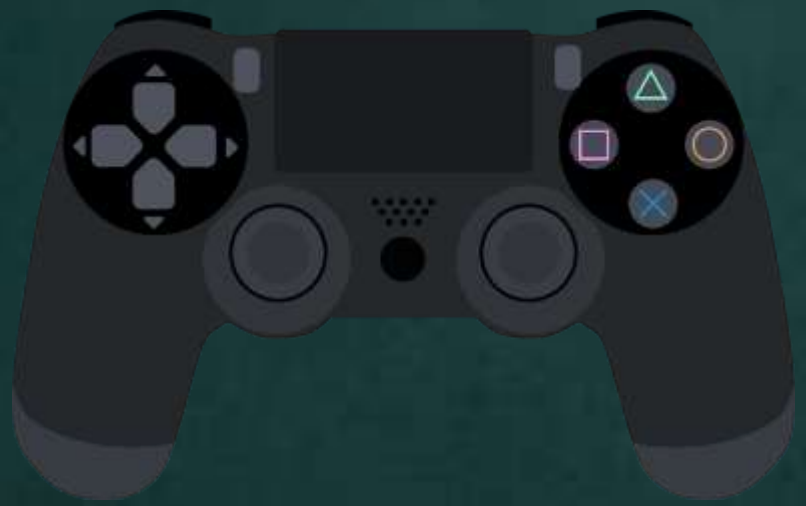
Data Output

	score bigint	dev_id character varying (10)	difficulty character varying (15)	ranks bigint
1	5300	bd_013	Difficult	1
2	4570	bd_013	Difficult	2
3	3370	bd_013	Difficult	3
4	5300	bd_015	Difficult	1
5	3200	bd_015	Low	2
6	1950	bd_015	Difficult	3
7	2400	bd_017	Low	1
8	1750	bd_017	Medium	2
9	390	bd_017	Low	3
10	2970	rf_013	Difficult	1
11	2700	rf_013	Medium	2
12	2300	rf_013	Medium	3
13	3950	rf_015	Difficult	1
14	2800	rf_015	Medium	2
15	900	rf_015	Medium	3
16	5140	rf_017	Medium	1
17	5140	rf_017	Difficult	2
18	3500	rf_017	Difficult	3
19	4390	wd_019	Difficult	1
20	1550	wd_019	Low	2

SQL Queries Analysis

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
```



	dev_id character varying (10)	time_stamp timestamp without time zone	rankss bigint
1	bd_013	2022-10-11 02:23:00	1
2	bd_015	2022-10-11 18:45:00	1
3	bd_017	2022-10-12 07:30:00	1
4	rf_013	2022-10-11 05:20:00	1
5	rf_015	2022-10-11 19:34:00	1
6	rf_017	2022-10-11 09:28:00	1
7	wd_019	2022-10-12 23:19:00	1
8	zm_013	2022-10-11 13:00:00	1
9	zm_015	2022-10-11 14:05:00	1
10	zm_017	2022-10-11 14:33:00	1

SQL Queries Analysis

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

	dev_id character varying (10)	score bigint	difficulty character varying (15)	ranks bigint
1	zm_017	5500	Difficult	1
2	zm_017	5500	Difficult	1
3	bd_013	5300	Difficult	3
4	bd_015	5300	Difficult	3
5	rf_017	5140	Difficult	5
6	zm_015	3470	Low	1
7	zm_017	3210	Low	2
8	bd_015	3200	Low	3
9	bd_013	2840	Low	4
10	zm_015	2800	Low	5
11	zm_017	5490	Medium	1
12	rf_017	5140	Medium	2
13	zm_015	4950	Medium	3
14	zm_015	4950	Medium	3
15	rf_015	2800	Medium	5

SQL Queries Analysis

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
```

Data Output

	p_id bigint	dev_id character varying (10)	time_stamp timestamp without time zone	rankss bigint
1	211	bd_017	2022-10-12 13:23:00	1
2	224	rf_017	2022-10-14 01:15:00	1
3	242	bd_013	2022-10-13 01:14:00	1
4	292	rf_013	2022-10-12 04:29:00	1
5	296	zm_017	2022-10-14 15:15:00	1
6	300	rf_013	2022-10-11 05:20:00	1
7	310	rf_017	2022-10-11 15:15:00	1
8	319	zm_017	2022-10-12 14:20:00	1
9	358	zm_017	2022-10-14 05:05:00	1
10	368	zm_015	2022-10-12 01:14:00	1
11	428	bd_015	2022-10-15 18:00:00	1
12	429	rf_017	2022-10-11 09:28:00	1
13	483	zm_017	2022-10-11 14:33:00	1
14	547	bd_013	2022-10-15 02:19:00	1
15	558	wd_019	2022-10-12 23:19:00	1
16	590	bd_017	2022-10-12 07:30:00	1
17	632	bd_013	2022-10-12 16:30:00	1
18	641	rf_015	2022-10-13 04:04:00	1
19	644	zm_015	2022-10-11 14:05:00	1
20	656	bd_013	2022-10-11 17:47:00	1

SQL Queries Analysis

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

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
```

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)
```

Data Output

	p_id bigint	dates date	kill_counts numeric
1	211	2022-10-12	45
2	211	2022-10-13	44
3	211	2022-10-14	9
4	211	2022-10-15	15
5	224	2022-10-14	54
6	224	2022-10-15	58
7	242	2022-10-13	21
8	242	2022-10-14	37
9	292	2022-10-12	21
10	292	2022-10-15	4
11	296	2022-10-14	11
12	300	2022-10-11	48
13	300	2022-10-12	18
14	300	2022-10-13	8
15	310	2022-10-11	20
16	310	2022-10-13	34
17	310	2022-10-15	14
18	319	2022-10-12	5
19	358	2022-10-14	7
20	368	2022-10-12	49

SQL Queries 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
```

Data Output

	p_id bigint	time_stamp timestamp without time zone	stages_crossed bigint	cumulative_of_stages_crossed numeric
1	211	2022-10-12 13:23:00	4	4
2	211	2022-10-12 18:30:00	5	9
3	211	2022-10-13 05:36:00	5	14
4	211	2022-10-13 22:30:00	5	19
5	211	2022-10-14 08:56:00	7	26
6	224	2022-10-14 01:15:00	7	7
7	224	2022-10-14 08:21:00	5	12
8	224	2022-10-15 05:30:00	10	22
9	242	2022-10-13 01:14:00	6	6
10	292	2022-10-12 04:29:00	4	4
11	296	2022-10-14 15:15:00	2	2
12	300	2022-10-11 05:20:00	7	7
13	300	2022-10-11 19:19:00	5	12
14	300	2022-10-12 01:45:00	2	14
15	300	2022-10-12 11:21:00	3	17
16	310	2022-10-11 15:15:00	7	7
17	310	2022-10-13 19:18:00	5	12
18	358	2022-10-14 05:05:00	3	3

SQL Queries Analysis

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
```

Data Output

	p_id bigint	dev_id character varying (10)	sum_ofscore numeric
1	224	bd_013	9870
2	310	bd_013	3370
3	211	bd_013	3200
4	310	bd_015	5300
5	683	bd_015	3200
6	368	bd_015	1950
7	590	bd_017	2400
8	644	bd_017	1750
9	211	bd_017	390
10	368	rf_013	2970
11	211	rf_013	2700
12	300	rf_013	2300
13	483	rf_015	3950
14	683	rf_015	2800
15	590	rf_015	900
16	310	rf_017	5140
17	224	rf_017	5140
18	429	rf_017	3500
19	483	wd_019	4390
20	590	wd_019	1550

SQL Queries Analysis

14. 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 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 ))
```

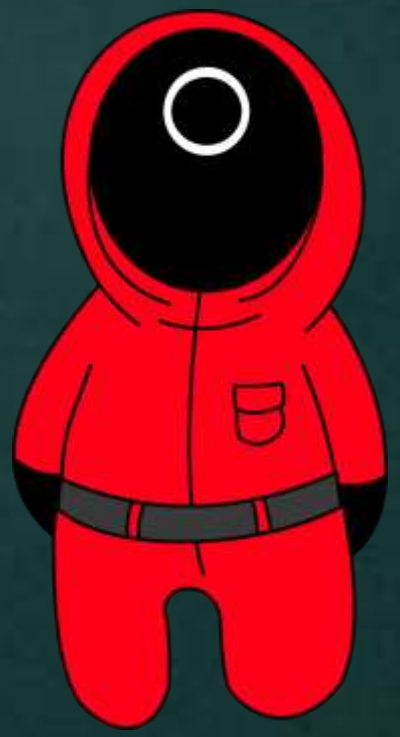
Data Output

	p_id bigint	sumofscore numeric
1	211	10940
2	224	16310
3	242	6310
4	300	4860
5	310	13810
6	368	8710
7	429	13220
8	483	17230
9	590	8000
10	632	10750

SQL Queries Analysis

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);
```



Dev_ID	Difficulty	Headshots_Count
bd_013	Medium	4
bd_013	Medium	8
bd_013	Medium	10
bd_015	Low	3
bd_015	Difficult	8
bd_015	Low	13
bd_017	Low	15
bd_017	Medium	16
bd_017	Low	18
rf_013	Low	3
rf_013	Medium	6
rf_013	Low	7
rf_015	Medium	0
rf_015	Medium	1
rf_015	Low	2
rf_017	Difficult	1
rf_017	Difficult	11
rf_017	Difficult	18
wd_019	Difficult	0
wd_019	Low	10
wd_019	Difficult	16
zm_013	Medium	1

Findings

- The Device_id “zm_017” has crossed maximum stages i.e., 6 at Difficulty level.
- P_id 483 on ’13-10-2022 6:20’ has max kill count i.e., 134
- Total 19 players have played games on multiple days.
- Average of kill count by L1 code :

  Bulls_eye = 22.2500

  Speed_Blitz = 19.3333

  War_Zone = 19.2857
- Lives earned by level’s :

  Level 1 = 23

  Level 2 = 51
- Top Score’s on Difficulty Level are 5300 and 4570.

Findings

By understanding player behaviour and preferences, developers can make data-driven decisions to improve game design, adjust difficulty levels, and create more engaging and challenging gameplay experiences.

Thank You