Decode Gaming Behaviour

1

Problem Statement

The main aspect of this project and the data asked.

2

Dataset Description

The datasets used in this project and their descriptions.

3

What we have to do?

The specific tasks asked and giving the results in the form of tables.

Problem Statement

In this internship, we will be working with a dataset related to a game. The dataset includes two tables: `Player Details` and `Level Details`.

Tools Used:
Microsoft SQL Server for writing queries asked in the tasks and giving the results in the form of a table.

Dataset:

- *Player Details*
- *Level Details*

From Mentorness

Dataset Description

The dataset includes two tables: `Player Details` and `Level Details`.
Below is a brief description of the dataset:
Player Details Table:

- *`P_ID`: Player ID*
- *`PName`: Player Name*
- *`L1_status`: Level 1 Status*
- *`L2_status`: Level 2 Status*
- *`L1_code`: Systemgenerated Level 1 Code*
- *`L2_code`: Systemgenerated Level 2 Code*

Level Details Table:

- *`P_ID`: Player ID*
- *`Dev_ID`: Device ID*
- *`start_datetime`: Start Time*
- *`stages_crossed`: Stages Crossed*
- *`level`: Game Level*
- *`difficulty`: Difficulty Level*
- *`kill_count`: Kill Count*
- *`headshots_count`: Headshots Count*
- *`score`: Player Score*
- *`lives_earned`: Extra Lives Earned*

What we need to do

Using the "Game Analysis.sql" file, there are 15 questions for which we have to find the answers by writing SQL queries.

1. *Extract `P_ID`, `Dev_ID`, `PName`, and `Difficulty_level` of all players at Level 0.*
2. *Find `Level1_code`wise average `Kill_Count` where `lives_earned` is 2, and at least 3 stages are crossed.*
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.*
4. *Extract `P_ID` and the total number of unique dates for those players who have played games on multiple days.*
5. *Find `P_ID` and levelwise sum of `kill_counts` where `kill_count` is greater than the average kill count for Medium difficulty.*
6. *Find `Level` and its corresponding `Level_code`wise sum of lives earned, excluding Level 0. Arrange in ascending order of 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.*
8. *Find the `first_login` datetime for each device ID.*
9. *Find the top 5 scores based on each difficulty level and rank them in increasing order using `Rank`. Display `Dev_ID` as well.*
10. *Find the device ID that is first logged in (based on `start_datetime`) for each player (`P_ID`).*

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

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

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

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

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

The SQL Query is below:

```
Select P_ID, Dev_ID, Difficulty
From level_details
Where Level=0;
```

The result from this query:

	P_ID	Dev_ID	Difficulty
1	656	rf_013	Medium
2	632	bd_013	Difficult
3	429	bd_013	Medium
4	310	bd_015	Difficult
5	211	bd_017	Low
6	300	zm_015	Difficult
7	358	zm_017	Low
8	358	zm_013	Medium
9	641	rf_013	Low
10	641	rf_015	Medium
11	641	rf_013	Difficult
12	558	wd_019	Difficult

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

The SQL Query is below:

```
Select L1_Code L1_code, AVG(Kill_Count) Avg_Kill_Count
From level_details
Join player_details On level_details.P_ID=player_details.P_ID
Where Lives_Earned=2 And Stages_crossed>=3
Group By L1_Code;
```

The result from this query:

	L1_code	Avg_Kill_Count
1	bulls_eye	22
2	speed_bli...	19
3	war_zone	19

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.

The SQL Query is below:

```
Select Difficulty, Sum(Stages_crossed) as Total_Stages_Crossed
From level_details
Where Level=2 And Dev_ID Like'%zm%'
Group By Difficulty
Order By Total_Stages_Crossed Desc;
```

The result from this query:

	Difficulty	Total_Stages_Cross...
1	Difficult	46
2	Medium	35
3	Low	15

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

The SQL Query is below:

```
Select P_ID, Count(Distinct start_datetime) as Unique_Dates
From level_details
Group By P_ID
Having Count(Distinct start_datetime)>1;
```

The result from this query:

	P_ID	Unique_Dates
1	211	6
2	224	4
3	242	2
4	292	2
5	296	2
6	300	5
7	310	3
8	358	2
9	368	4
10	429	4
11	483	5
12	547	3
13	590	5
14	632	5
15	641	3
16	644	3
17	656	4
18	663	5
19	683	7

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

The SQL Query is below:

```
Select P_ID, Level, Sum(Kill_Count) as Total_Kill_Count
From level_details
Where Kill_Count>
(Select Avg(Kill_Count)
From level_details
Where Difficulty Like 'Medium')
Group By Level, P_ID;
```

The result from this query:

	P_ID	Level	Total_Kill_Count
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
21	632	2	53
22	644	2	24
23	656	1	37
24	663	1	73
25	663	2	53
26	683	1	21
27	683	2	64

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

The SQL Query is below:

```
Select Level, L1_Code,L2_Code,Sum(Lives_Earned) As
Total_Lives_Earned
From level_details
Join player_details
On level_details.P_ID=player_details.P_ID
Where Level<>0
Group By Level, L1_Code, L2_Code
Order By Level Asc;
```

The result from this query:

	Level	L1_Code	L2_Code	Total_Lives_Earned
1	1	bulls_eye	NULL	3
2	1	bulls_eye	cosmic_vision	1
3	1	bulls_eye	resurgence	1
4	1	leap_of_fai...	NULL	0
5	1	speed_blitz	NULL	0
6	1	speed_blitz	cosmic_vision	4
7	1	speed_blitz	splippery_slo...	3
8	1	war_zone	NULL	4
9	1	war_zone	resurgence	0
10	1	war_zone	splippery_slo...	7
11	2	bulls_eye	cosmic_vision	6
12	2	bulls_eye	resurgence	8
13	2	speed_blitz	cosmic_vision	6
14	2	speed_blitz	splippery_slo...	14
15	2	war_zone	resurgence	3
16	2	war_zone	splippery_slo...	14

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.

The SQL Query is below:

```
With Ranked_Scores as
(Select *, ROW_NUMBER() Over (Partition By Dev_ID Order By Score
Desc) as Rank
From level_details
)
Select Dev_ID, Score, Difficulty
From Ranked_Scores
Where Rank<=3;
```

The result from this query:

	Dev_ID	Score	Difficulty
1	bd_013	5300	Difficult
2	bd_013	4570	Difficult
3	bd_013	3370	Difficult
4	bd_015	5300	Difficult
5	bd_015	3200	Low
6	bd_015	1950	Difficult
7	bd_017	2400	Low
8	bd_017	1750	Medium
9	bd_017	390	Low
10	rf_013	2970	Difficult
11	rf_013	2700	Medium
12	rf_013	2300	Medium
13	rf_015	3950	Difficult
14	rf_015	2800	Medium
15	rf_015	900	Medium
16	rf_017	5140	Difficult
17	rf_017	5140	Medium
18	rf_017	3500	Difficult
19	wd_019	4390	Difficult
20	wd_019	1550	Low
21	wd_019	635	Difficult
22	zm_013	4710	Difficult
23	zm_013	2350	Medium
24	zm_013	120	Medium
25	zm_015	4950	Medium
26	zm_015	4950	Medium
27	zm_015	3470	Low
28	zm_017	5500	Difficult
29	zm_017	5500	Difficult
30	zm_017	5490	Medium

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

The SQL Query is below:

```
Select Dev_ID, Min(start_datetime) as First_Login
From level_details
Group By Dev_ID;
```

The result from this query:

	Dev_ID	First_Login
1	bd_013	2022-10-11 02:23:45.000
2	bd_015	2022-10-11 18:45:55.000
3	bd_017	2022-10-12 07:30:18.000
4	rf_013	2022-10-11 05:20:40.000
5	rf_015	2022-10-11 19:34:25.000
6	rf_017	2022-10-11 09:28:56.000
7	wd_019	2022-10-12 23:19:17.000
8	zm_013	2022-10-11 13:00:22.000
9	zm_015	2022-10-11 14:05:08.000
10	zm_017	2022-10-11 14:33:27.000

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

The SQL Query is below:

```
With Ranked_Scores as
(Select *, Rank() Over (Partition By Difficulty Order By Score
Desc) as Rank
From level_details
)
Select Dev_ID, Score, Difficulty
From Ranked_Scores
Where Rank<=5;
```

The result from this query:

	Dev_ID	Score	Difficulty
1	zm_017	5500	Difficult
2	zm_017	5500	Difficult
3	bd_013	5300	Difficult
4	bd_015	5300	Difficult
5	rf_017	5140	Difficult
6	zm_015	3470	Low
7	zm_017	3210	Low
8	bd_015	3200	Low
9	bd_013	2840	Low
10	zm_015	2800	Low
11	zm_017	5490	Medium
12	rf_017	5140	Medium
13	zm_015	4950	Medium
14	zm_015	4950	Medium
15	rf_015	2800	Medium

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.

The SQL Query is below:

```
Select P_ID, Dev_ID, Min(start_datetime) as First_Login
From level_details
Group By P_ID, Dev_ID;
```

The result from this query:

	P_ID	Dev_ID	First_Login
1	211	bd_013	2022-10-12 18:30:30.0...
2	224	bd_013	2022-10-15 05:30:28.0...
3	242	bd_013	2022-10-13 01:14:29.0...
4	300	bd_013	2022-10-11 19:19:19.0...
5	310	bd_013	2022-10-15 23:30:50.0...
6	429	bd_013	2022-10-11 19:28:43.0...
7	547	bd_013	2022-10-15 02:19:27.0...
8	632	bd_013	2022-10-12 16:30:30.0...
9	656	bd_013	2022-10-11 17:47:09.0...
10	663	bd_013	2022-10-15 17:30:30.0...
11	683	bd_013	2022-10-11 02:23:45.0...
12	224	bd_015	2022-10-14 08:21:49.0...
13	310	bd_015	2022-10-13 19:18:20.0...
14	368	bd_015	2022-10-12 11:59:18.0...
15	428	bd_015	2022-10-15 18:00:00.0...
16	483	bd_015	2022-10-11 22:20:10.0...
17	656	bd_015	2022-10-13 22:19:45.0...
18	683	bd_015	2022-10-11 18:45:55.0...
19	211	bd_017	2022-10-12 13:23:45.0...
20	590	bd_017	2022-10-12 07:30:18.0...
21	644	bd_017	2022-10-12 23:52:18.0...
22	211	rf_013	2022-10-13 05:36:15.0...
23	292	rf_013	2022-10-12 04:29:45.0...
24	300	rf_013	2022-10-11 05:20:40.0...
25	368	rf_013	2022-10-15 14:47:53.0...
26	547	rf_013	2022-10-15 07:15:15.0...
27	590	rf_013	2022-10-12 19:23:15.0...
28	632	rf_013	2022-10-12 19:36:40.0...
29	641	rf_013	2022-10-14 01:25:30.0...
30	656	rf_013	2022-10-15 18:12:50.0...

11. For each player and date, determine how many `kill_counts` were played by the player so far.
a) Using window functions

The SQL Query is below:

```
Select P_ID, start_datetime,
Sum(Kill_Count) Over (Partition By P_ID Order By start_datetime)
as kill_count_played_so_far
From level_details;
```

The result from this query:

	P_ID	start_datetime	kill_count_played_so_f...
1	211	2022-10-12 13:23:45.0...	20
2	211	2022-10-12 18:30:30.0...	45
3	211	2022-10-13 05:36:15.0...	75
4	211	2022-10-13 22:30:18.0...	89
5	211	2022-10-14 08:56:24.0...	98
6	211	2022-10-15 11:41:19.0...	113
7	224	2022-10-14 01:15:56.0...	20
8	224	2022-10-14 08:21:49.0...	54
9	224	2022-10-15 05:30:28.0...	84
10	224	2022-10-15 13:43:50.0...	112
11	242	2022-10-13 01:14:29.0...	21
12	242	2022-10-14 04:38:50.0...	58
13	292	2022-10-12 04:29:45.0...	21
14	292	2022-10-15 10:19:30.0...	25
15	296	2022-10-14 15:15:15.0...	7
16	296	2022-10-14 19:35:49.0...	11
17	300	2022-10-11 05:20:40.0...	23
18	300	2022-10-11 19:19:19.0...	48
19	300	2022-10-12 01:45:17.0...	52
20	300	2022-10-12 11:21:20.0...	66
21	300	2022-10-13 23:15:42.0...	74
22	310	2022-10-11 15:15:15.0...	20
23	310	2022-10-13 19:18:20.0...	54
24	310	2022-10-15 23:30:50.0...	68
25	319	2022-10-12 14:20:40.0...	5
26	358	2022-10-14 05:05:05.0...	4
27	358	2022-10-14 18:23:29.0...	7
28	368	2022-10-12 01:14:34.0...	20
29	368	2022-10-12 04:20:30.0...	34
30	368	2022-10-12 11:59:18.0...	49

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

The SQL Query is below:

```
Select P_ID, start_datetime,
    (Select Sum(Kill_Count) From level_details L1 Where
L2.P_ID=L1.P_ID and L1.start_datetime<=L2.start_datetime) AS
kill_count_played_so_far
From level_details L2
Order By P_ID, start_datetime;
```

The result from this query:

	P_ID	start_datetime	kill_count_played_so_f...
1	211	2022-10-12 13:23:45.0...	20
2	211	2022-10-12 18:30:30.0...	45
3	211	2022-10-13 05:36:15.0...	75
4	211	2022-10-13 22:30:18.0...	89
5	211	2022-10-14 08:56:24.0...	98
6	211	2022-10-15 11:41:19.0...	113
7	224	2022-10-14 01:15:56.0...	20
8	224	2022-10-14 08:21:49.0...	54
9	224	2022-10-15 05:30:28.0...	84
10	224	2022-10-15 13:43:50.0...	112
11	242	2022-10-13 01:14:29.0...	21
12	242	2022-10-14 04:38:50.0...	58
13	292	2022-10-12 04:29:45.0...	21
14	292	2022-10-15 10:19:30.0...	25
15	296	2022-10-14 15:15:15.0...	7
16	296	2022-10-14 19:35:49.0...	11
17	300	2022-10-11 05:20:40.0...	23
18	300	2022-10-11 19:19:19.0...	48
19	300	2022-10-12 01:45:17.0...	52
20	300	2022-10-12 11:21:20.0...	66
21	300	2022-10-13 23:15:42.0...	74
22	310	2022-10-11 15:15:15.0...	20
23	310	2022-10-13 19:18:20.0...	54
24	310	2022-10-15 23:30:50.0...	68
25	319	2022-10-12 14:20:40.0...	5
26	358	2022-10-14 05:05:05.0...	4
27	358	2022-10-14 18:23:29.0...	7
28	368	2022-10-12 01:14:34.0...	20
29	368	2022-10-12 04:20:30.0...	34
30	368	2022-10-12 11:59:18.0...	49

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

The SQL Query is below:

```
With excluded_latest_start As (
    Select P_ID, Max(start_datetime) As latest_start_datetime
    From level_details
    Group By P_ID),
cumulative_sum As (
    Select L.P_ID, L.start_datetime,
           Sum(Stages_crossed) Over (Partition By L.P_ID Order By
L.start_datetime) As stages_crossed
    From level_details L
        Join excluded_latest_start E On L.P_ID = E.P_ID
    Where E.latest_start_datetime Is Null Or L.start_datetime<
E.latest_start_datetime)
Select P_ID, start_datetime, stages_crossed As
cumulative_stages_crossed
From cumulative_sum;
```

The result from this query:

	P_ID	start_datetime	cumulative_stages_cross...
1	211	2022-10-12 13:23:45.0...	4
2	211	2022-10-12 18:30:30.0...	9
3	211	2022-10-13 05:36:15.0...	14
4	211	2022-10-13 22:30:18.0...	19
5	211	2022-10-14 08:56:24.0...	26
6	224	2022-10-14 01:15:56.0...	7
7	224	2022-10-14 08:21:49.0...	12
8	224	2022-10-15 05:30:28.0...	22
9	242	2022-10-13 01:14:29.0...	6
10	292	2022-10-12 04:29:45.0...	4
11	296	2022-10-14 15:15:15.0...	2
12	300	2022-10-11 05:20:40.0...	7
13	300	2022-10-11 19:19:19.0...	12
14	300	2022-10-12 01:45:17.0...	14
15	300	2022-10-12 11:21:20.0...	17
16	310	2022-10-11 15:15:15.0...	7
17	310	2022-10-13 19:18:20.0...	12
18	358	2022-10-14 05:05:05.0...	3
19	368	2022-10-12 01:14:34.0...	7
20	368	2022-10-12 04:20:30.0...	12
21	368	2022-10-12 11:59:18.0...	18
22	429	2022-10-11 09:28:56.0...	2
23	429	2022-10-11 13:00:22.0...	9
24	429	2022-10-11 19:28:43.0...	15
25	483	2022-10-11 14:33:27.0...	10
26	483	2022-10-11 22:20:10.0...	15
27	483	2022-10-12 02:40:20.0...	22
28	483	2022-10-12 19:30:11.0...	25
29	547	2022-10-15 02:19:27.0...	8
30	547	2022-10-15 07:15:15.0...	10

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

The SQL Query is below:

```
With Ranked_Scores as
(Select *, Rank() Over (Partition By Dev_ID Order By Score Desc)
as Rank
From level_details
)
Select Dev_ID, Score, P_ID
From Ranked_Scores
Where Rank<=3;
```

The result from this query:

	Dev_ID	Score	P_ID
1	bd_013	5300	224
2	bd_013	4570	224
3	bd_013	3370	310
4	bd_015	5300	310
5	bd_015	3200	683
6	bd_015	1950	368
7	bd_017	2400	590
8	bd_017	1750	644
9	bd_017	390	211
10	rf_013	2970	368
11	rf_013	2700	211
12	rf_013	2300	300
13	rf_015	3950	483
14	rf_015	2800	683
15	rf_015	900	590
16	rf_017	5140	310
17	rf_017	5140	224
18	rf_017	3500	429
19	wd_019	4390	483
20	wd_019	1550	590
21	wd_019	635	558
22	zm_013	4710	429
23	zm_013	2350	483
24	zm_013	120	358
25	zm_015	4950	663
26	zm_015	4950	632
27	zm_015	3470	242
28	zm_017	5500	632
29	zm_017	5500	663
30	zm_017	5490	483

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

The SQL Query is below:

```
Select P_ID,Score
From level_details
Group By P_ID, Score
Having Score>(Select Avg(Total_Score) * 0.5 From (Select
Sum(Score) as Total_Score from level_details Group By P_ID) as
avg_score)
Order By P_ID
```

The result from this query:

	P_ID	Score
1	224	4570
2	224	5140
3	224	5300
4	310	5140
5	310	5300
6	429	4710
7	483	3950
8	483	4390
9	483	5490
10	632	4950
11	632	5500
12	663	4950
13	663	5500
14	683	4100

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.

The SQL Query is below:

```
CREATE PROCEDURE GetTopNHeadshotsWithDifficulty
    @n INT
AS
BEGIN
    SELECT dev_id, headshots_count, difficulty,
           ROW_NUMBER() OVER (PARTITION BY dev_id ORDER BY
headshots_count DESC) AS rank
    FROM (
        SELECT Dev_ID, Headshots_Count, Difficulty,
               ROW_NUMBER() OVER (PARTITION BY Dev_ID ORDER BY
Headshots_Count DESC) AS rnk
        FROM level_details
    ) AS ranked
    WHERE rnk <= @n
END
Exec GetTopNHeadshotsWithDifficulty @n=5
```

The result from this query:

	dev_id	headshots_count	difficulty	rank
1	bd_013	30	Difficult	1
2	bd_013	30	Difficult	2
3	bd_013	25	Difficult	3
4	bd_013	22	Difficult	4
5	bd_013	17	Low	5
6	bd_015	30	Difficult	1
7	bd_015	30	Difficult	2
8	bd_015	20	Low	3
9	bd_015	17	Medium	4
10	bd_015	13	Low	5
11	bd_017	18	Low	1
12	bd_017	16	Medium	2
13	bd_017	15	Low	3
14	rf_013	25	Medium	1
15	rf_013	25	Medium	2
16	rf_013	19	Difficult	3
17	rf_013	17	Low	4
18	rf_013	17	Medium	5
19	rf_015	18	Medium	1
20	rf_015	10	Difficult	2
21	rf_015	3	Medium	3
22	rf_015	2	Low	4
23	rf_015	1	Medium	5
24	rf_017	27	Difficult	1
25	rf_017	18	Difficult	2
26	rf_017	18	Medium	3
27	rf_017	11	Difficult	4
28	rf_017	1	Difficult	5
29	wd_019	19	Difficult	1
30	wd_019	16	Difficult	2
31	wd_019	10	Low	3
32	wd_019	0	Difficult	4
33	zm_0...	20	Difficult	1
34	zm_0...	10	Medium	2
35	zm_0...	1	Medium	3
36	zm_0...	26	Low	1
37	zm_0...	20	Medium	2
38	zm_0...	20	Medium	3
39	zm_0...	18	Medium	4
40	zm_0...	9	Low	5
41	zm_0...	43	Medium	1
42	zm_0...	24	Difficult	2
43	zm_0...	24	Difficult	3
44	zm_0...	18	Low	4
45	zm_0...	18	Difficult	5

Conclusion

- Our Project harnessed Game Analysis data, leveraging SQL for robust database management. We meticulously prepared and analysed the data using SQL, addressing 15 key problem statements.
- These insights, encompassing player profile, performance and retention strategies. Our project emphasizes on managing large datasets and systematic analysis.
- The outcomes of this project hold significant potential to inform future strategies and drive decision-making, showcasing the value of rigorous data analysis with SQL environment.