GAME ANALYSIS USING SQL

Decode Gaming Behavior

Dataset Description

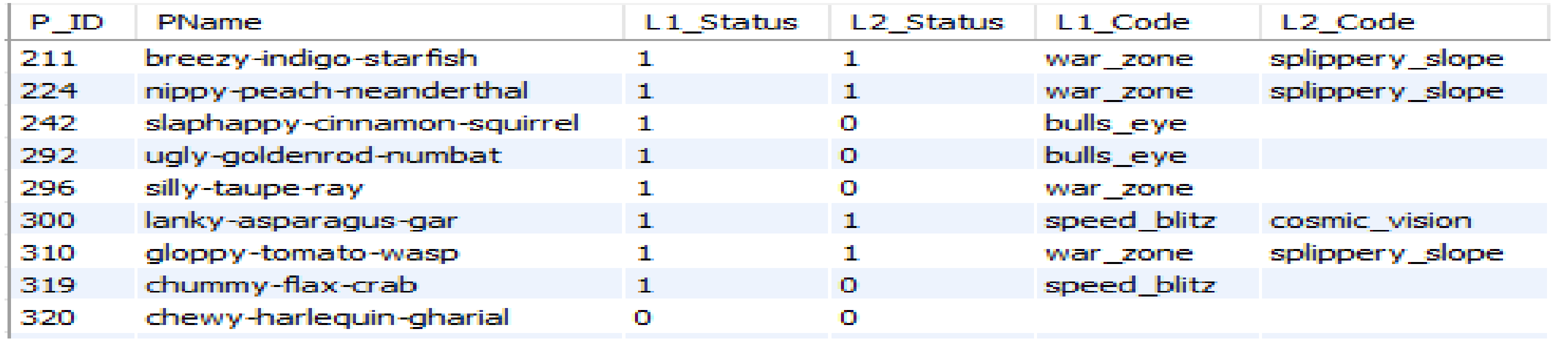
Player Details Table:

- \`P_ID\`: Player ID
- \`PName\`: 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:

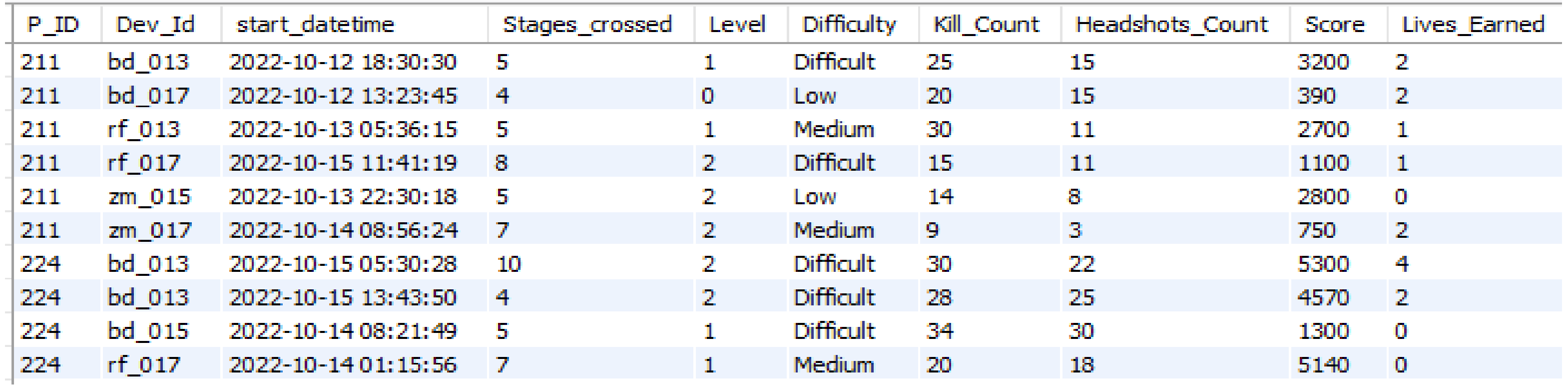
- \`P_ID\`: Player ID
- \`Dev_ID\`: Device ID
- \`start_time\`: Start Time
- \`stages_crossed\`: Stages Crossed
- \`level\`: Game Level
- \`difficulty\`: Difficulty Level
- \`kill_count\`: Kill Count
- \`headshots_count\`: Headshots Count
- \`score\`: Player Score
- \`lives_earned\`: Extra Lives Earned

Player Details Table



P_ID	PName	L1_Status	L2_Status	L1_Code	L2_Code
211	breezy-indigo-starfish	1	1	war_zone	splippery_slope
224	nippy-peach-neanderthal	1	1	war_zone	splippery_slope
242	slaphappy-cinnamon-squirrel	1	0	bulls_eye	
292	ugly-goldenrod-numbat	1	0	bulls_eye	
296	silly-taupe-ray	1	0	war_zone	
300	lanky-asparagus-gar	1	1	speed_blitz	cosmic_vision
310	gloppy-tomato-wasp	1	1	war_zone	splippery_slope
319	chummy-flax-crab	1	0	speed_blitz	
320	chewy-harlequin-gharial	0	0		

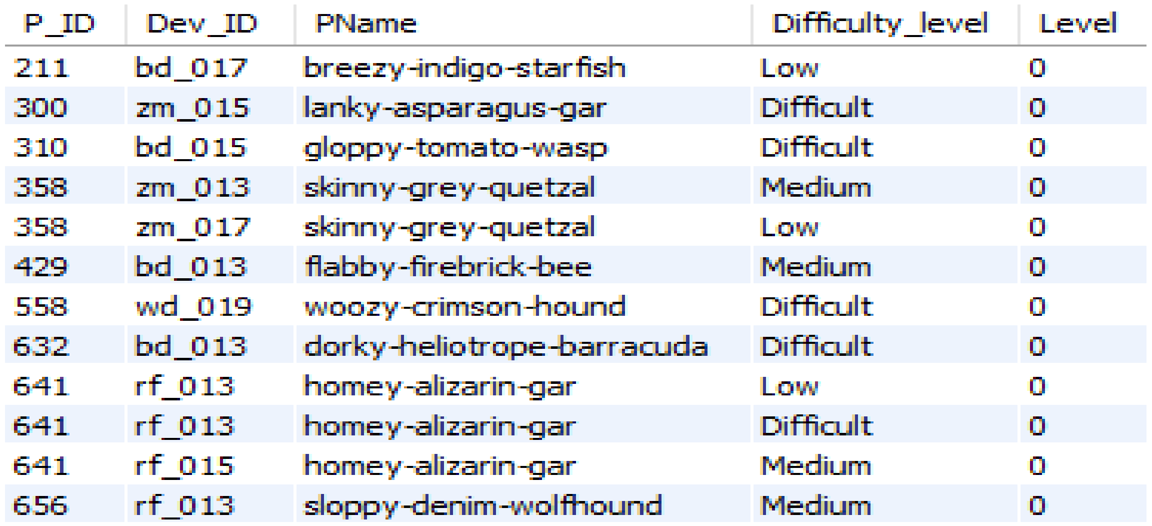
Level Details Table



P_ID	Dev_Id	start_datetime	Stages_crossed	Level	Difficulty	Kill_Count	Headshots_Count	Score	Lives_Earned
211	bd_013	2022-10-12 18:30:30	5	1	Difficult	25	15	3200	2
211	bd_017	2022-10-12 13:23:45	4	0	Low	20	15	390	2
211	rf_013	2022-10-13 05:36:15	5	1	Medium	30	11	2700	1
211	rf_017	2022-10-15 11:41:19	8	2	Difficult	15	11	1100	1
211	zm_015	2022-10-13 22:30:18	5	2	Low	14	8	2800	0
211	zm_017	2022-10-14 08:56:24	7	2	Medium	9	3	750	2
224	bd_013	2022-10-15 05:30:28	10	2	Difficult	30	22	5300	4
224	bd_013	2022-10-15 13:43:50	4	2	Difficult	28	25	4570	2
224	bd_015	2022-10-14 08:21:49	5	1	Difficult	34	30	1300	0
224	rf_017	2022-10-14 01:15:56	7	1	Medium	20	18	5140	0

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

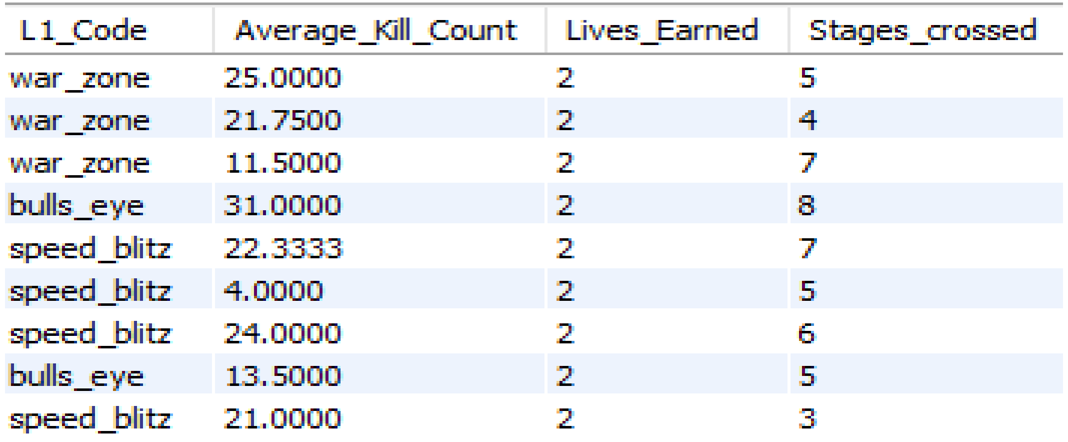
```sql
SELECT p.P_ID, ld.Dev_ID, p.PName, ld.Difficulty AS Difficulty_level,Level
FROM player_details p
JOIN level_details2 ld
ON p.P_ID = ld.P_ID
WHERE ld.Level = 0;
```



P_ID	Dev_ID	PName	Difficulty_level	Level
211	bd_017	breezy-indigo-starfish	Low	0
300	zm_015	lanky-asparagus-gar	Difficult	0
310	bd_015	gloppy-tomato-wasp	Difficult	0
358	zm_013	skinny-grey-quetzal	Medium	0
358	zm_017	skinny-grey-quetzal	Low	0
429	bd_013	flabby-firebrick-bee	Medium	0
558	wd_019	woozy-crimson-hound	Difficult	0
632	bd_013	dorky-heliotrope-barracuda	Difficult	0
641	rf_013	homey-alizarin-gar	Low	0
641	rf_013	homey-alizarin-gar	Difficult	0
641	rf_015	homey-alizarin-gar	Medium	0
656	rf_013	sloppy-denim-wolfhound	Medium	0

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

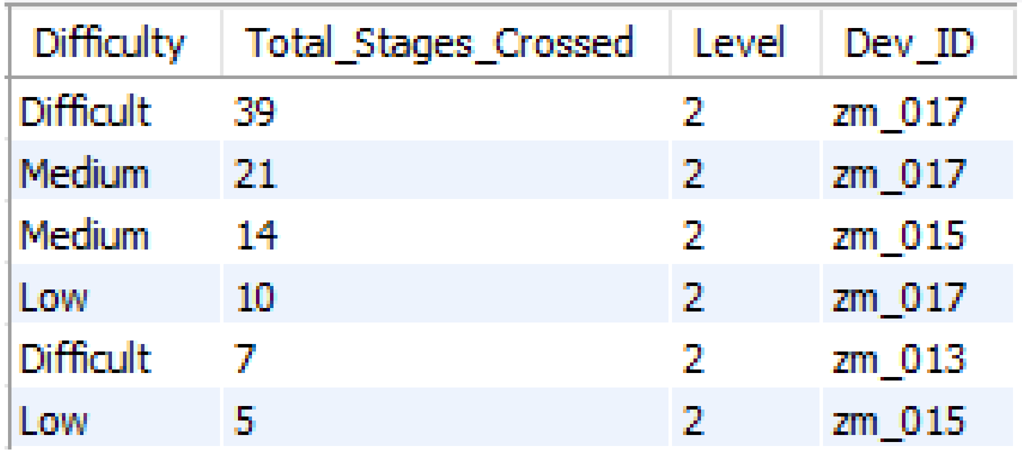
```
SELECT pd.L1_Code, AVG(ld.Kill_Count) AS Average_Kill_Count,
ld.Lives_Earned, ld.Stages_crossed
FROM player_details pdJOIN level_details2 ld
ON pd.P_ID = ld.P_ID
WHERE ld.Lives_Earned = 2 AND ld.Stages_crossed >= 3
GROUP BY pd.L1_Code, ld.Lives_Earned, ld.Stages_crossed;
```



L1_Code	Average_Kill_Count	Lives_Earned	Stages_crossed
war_zone	25.0000	2	5
war_zone	21.7500	2	4
war_zone	11.5000	2	7
bulls_eye	31.0000	2	8
speed_blitz	22.3333	2	7
speed_blitz	4.0000	2	5
speed_blitz	24.0000	2	6
bulls_eye	13.5000	2	5
speed_blitz	21.0000	2	3

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.

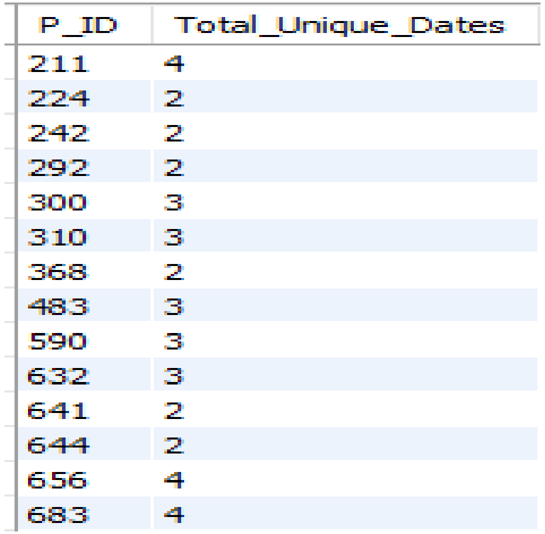
```
SELECT ld.Difficulty, SUM(ld.Stages_crossed) AS Total_Stages_Crossed, Level, ld.Dev_ID
FROM level_details2 ld
JOIN player_details pd
ON ld.P_ID = pd.P_IDWHERE ld.Level = 2 AND ld.Dev_ID LIKE 'zm_%‘
GROUP BY ld.Difficulty, ld.Dev_ID
ORDER BY Total_Stages_Crossed DESC;
```



Difficulty	Total_Stages_Crossed	Level	Dev_ID
Difficult	39	2	zm_017
Medium	21	2	zm_017
Medium	14	2	zm_015
Low	10	2	zm_017
Difficult	7	2	zm_013
Low	5	2	zm_015

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

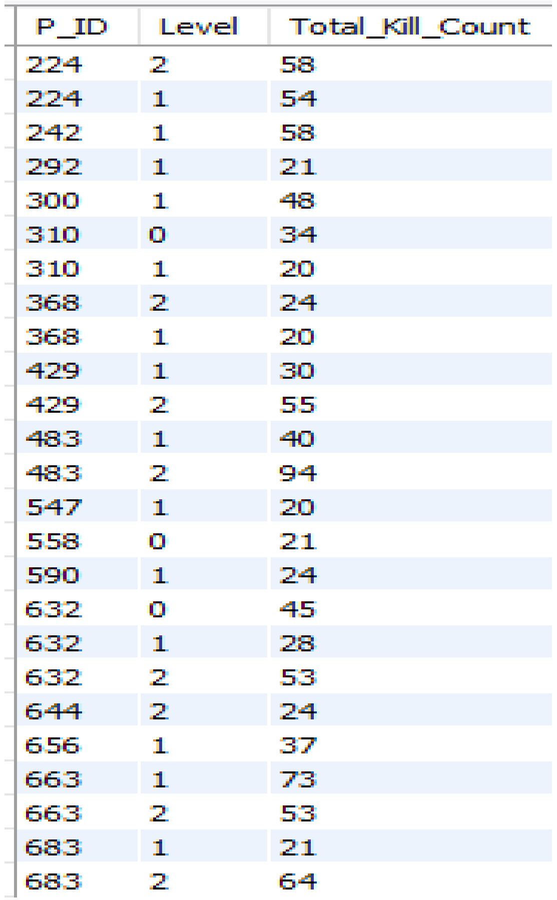
```
SELECT P_ID, COUNT(DISTINCT DATE_FORMAT(start_datetime, %y-%m-%d'))
AS Total_Unique_Dates
FROM level_details2
GROUP BY P_ID
HAVING COUNT(DISTINCT DATE_FORMAT(start_datetime, '%y-%m-%d')) > 1;
```



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

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

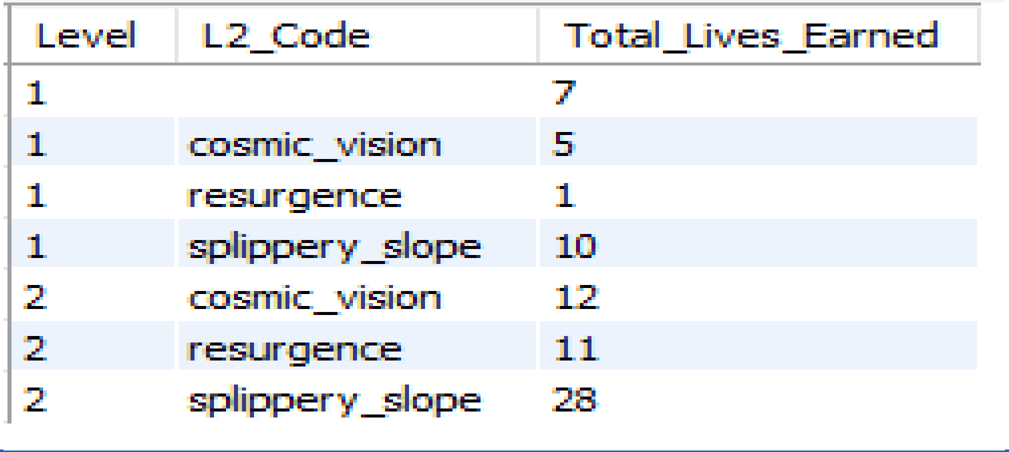
```
SELECT P_ID, Level,
SUM(Kill_Count) AS Total_Kill_Count
FROM level_details2
WHERE Kill_Count > (
                    SELECT AVG(Kill_Count)
                    FROM level_details2
                    WHERE Difficulty = 'Medium‘
)
GROUP BY P_ID, Level;
```



P_ID	Level	Total_Kill_Count
224	2	58
224	1	54
242	1	58
292	1	21
300	1	48
310	0	34
310	1	20
368	2	24
368	1	20
429	1	30
429	2	55
483	1	40
483	2	94
547	1	20
558	0	21
590	1	24
632	0	45
632	1	28
632	2	53
644	2	24
656	1	37
663	1	73
663	2	53
683	1	21
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.

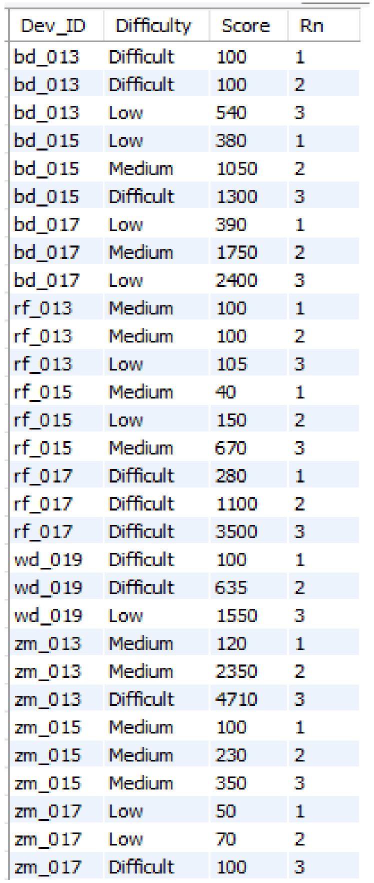
```
SELECT ld.Level, pd.L2_Code, SUM(ld.Lives_Earned) AS Total_Lives_Earned
FROM level_details2 ld
JOIN player_details pd
ON ld.P_ID = pd.P_ID
WHERE ld.Level > 0
GROUP BY ld.Level, pd.L2_Code
ORDER BY ld.Level ASC;
```



Level	L2_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

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.

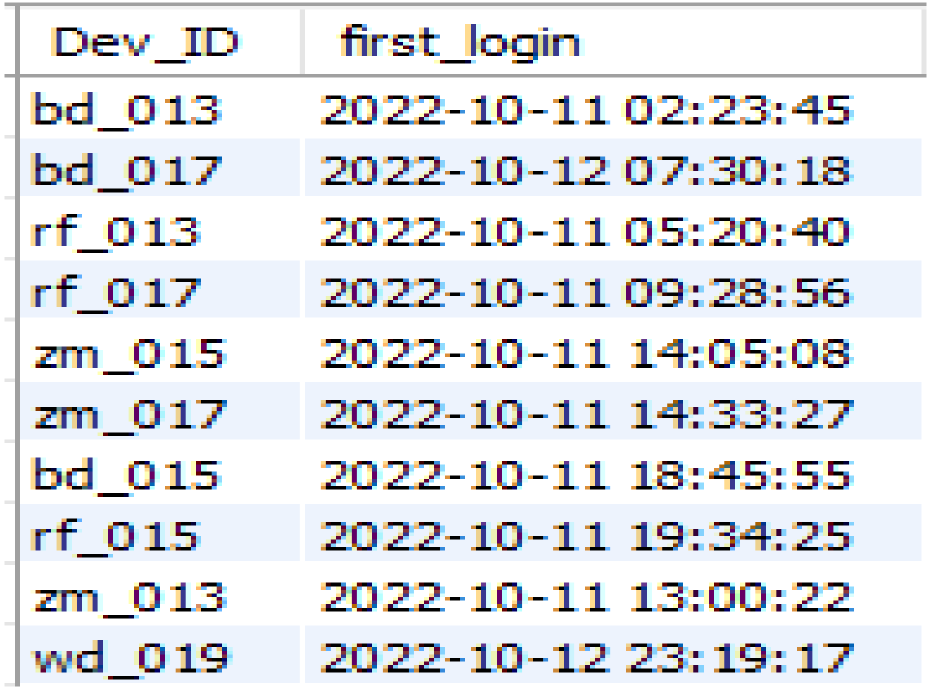
```sql
SELECT subquery.Dev_ID, subquery.Difficulty, subquery.Score,
subquery.Rn
FROM (
SELECT ld.Dev_ID, ld.Difficulty, ld.Score,
ROW_NUMBER( )
OVER(PARTITION BY ld.Dev_ID ORDER BY ld.Score) AS Rn
FROM level_details2 ld  ) AS subquery
WHERE Rn <= 3;
```



Dev_ID	Difficulty	Score	Rn
bd_013	Difficult	100	1
bd_013	Difficult	100	2
bd_013	Low	540	3
bd_015	Low	380	1
bd_015	Medium	1050	2
bd_015	Difficult	1300	3
bd_017	Low	390	1
bd_017	Medium	1750	2
bd_017	Low	2400	3
rf_013	Medium	100	1
rf_013	Medium	100	2
rf_013	Low	105	3
rf_015	Medium	40	1
rf_015	Low	150	2
rf_015	Medium	670	3
rf_017	Difficult	280	1
rf_017	Difficult	1100	2
rf_017	Difficult	3500	3
wd_019	Difficult	100	1
wd_019	Difficult	635	2
wd_019	Low	1550	3
zm_013	Medium	120	1
zm_013	Medium	2350	2
zm_013	Difficult	4710	3
zm_015	Medium	100	1
zm_015	Medium	230	2
zm_015	Medium	350	3
zm_017	Low	50	1
zm_017	Low	70	2
zm_017	Difficult	100	3

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

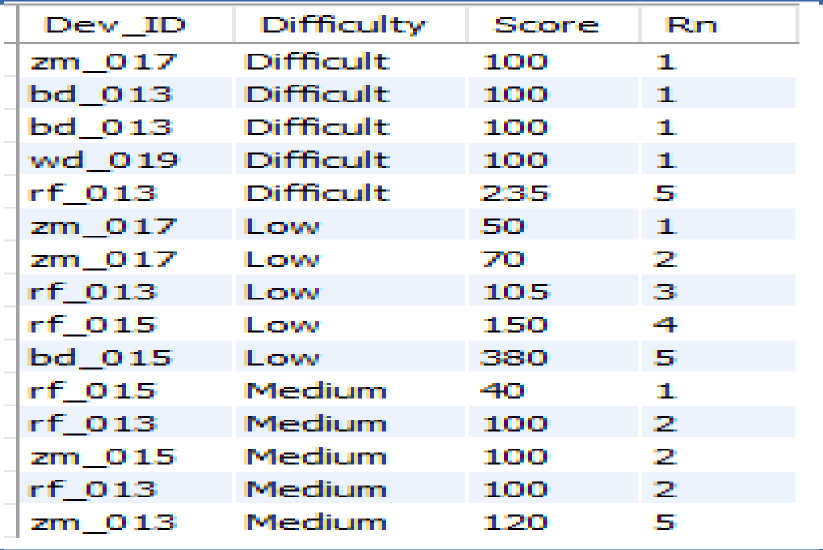
```
SELECT Dev_ID, MIN(start_datetime) AS first_login
FROM level_details2
GROUP BY Dev_ID;
```



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

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

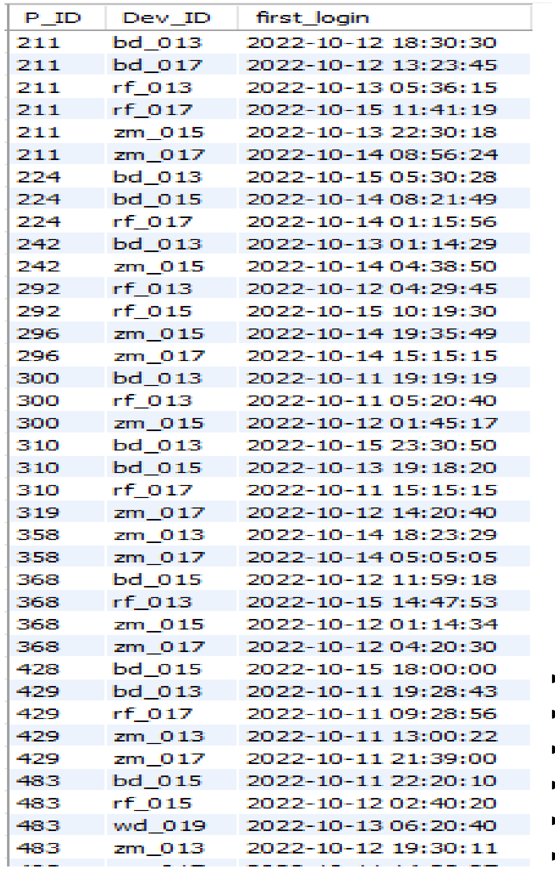
```
SELECT subquery.Dev_ID, subquery.Difficulty, subquery.Score, subquery.Rn
FROM (
     SELECT ld.Dev_ID, ld.Difficulty, ld.Score,
    RANK( ) OVER (PARTITION BY ld.Difficulty ORDER BY ld.Score ASC) AS Rn
    FROM level_details2 ld) AS subquery
WHERE Rn <= 5;
```



Dev_ID	Difficulty	Score	Rn
zm_017	Difficult	100	1
bd_013	Difficult	100	1
bd_013	Difficult	100	1
wd_019	Difficult	100	1
rf_013	Difficult	235	5
zm_017	Low	50	1
zm_017	Low	70	2
rf_013	Low	105	3
rf_015	Low	150	4
bd_015	Low	380	5
rf_015	Medium	40	1
rf_013	Medium	100	2
zm_015	Medium	100	2
rf_013	Medium	100	2
zm_013	Medium	120	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, MIN(start_datetime) AS first_login
FROM level_details2
GROUP BY P_ID, Dev_ID;
```



P_ID	Dev_ID	first_login
211	bd_013	2022-10-12 18:30:30
211	bd_017	2022-10-12 13:23:45
211	rf_013	2022-10-13 05:36:15
211	rf_017	2022-10-15 11:41:19
211	zm_015	2022-10-13 22:30:18
211	zm_017	2022-10-14 08:56:24
224	bd_013	2022-10-15 05:30:28
224	bd_015	2022-10-14 08:21:49
224	rf_017	2022-10-14 01:15:56
242	bd_013	2022-10-13 01:14:29
242	zm_015	2022-10-14 04:38:50
292	rf_013	2022-10-12 04:29:45
292	rf_015	2022-10-15 10:19:30
296	zm_015	2022-10-14 19:35:49
296	zm_017	2022-10-14 15:15:15
300	bd_013	2022-10-11 19:19:19
300	rf_013	2022-10-11 05:20:40
300	zm_015	2022-10-12 01:45:17
310	bd_013	2022-10-15 23:30:50
310	bd_015	2022-10-13 19:18:20
310	rf_017	2022-10-11 15:15:15
319	zm_017	2022-10-12 14:20:40
358	zm_013	2022-10-14 18:23:29
358	zm_017	2022-10-14 05:05:05
368	bd_015	2022-10-12 11:59:18
368	rf_013	2022-10-15 14:47:53
368	zm_015	2022-10-12 01:14:34
368	zm_017	2022-10-12 04:20:30
428	bd_015	2022-10-15 18:00:00
429	bd_013	2022-10-11 19:28:43
429	rf_017	2022-10-11 09:28:56
429	zm_013	2022-10-11 13:00:22
429	zm_017	2022-10-11 21:39:00
483	bd_015	2022-10-11 22:20:10
483	rf_015	2022-10-12 02:40:20
483	wd_019	2022-10-13 06:20:40
483	zm_013	2022-10-12 19:30:11

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

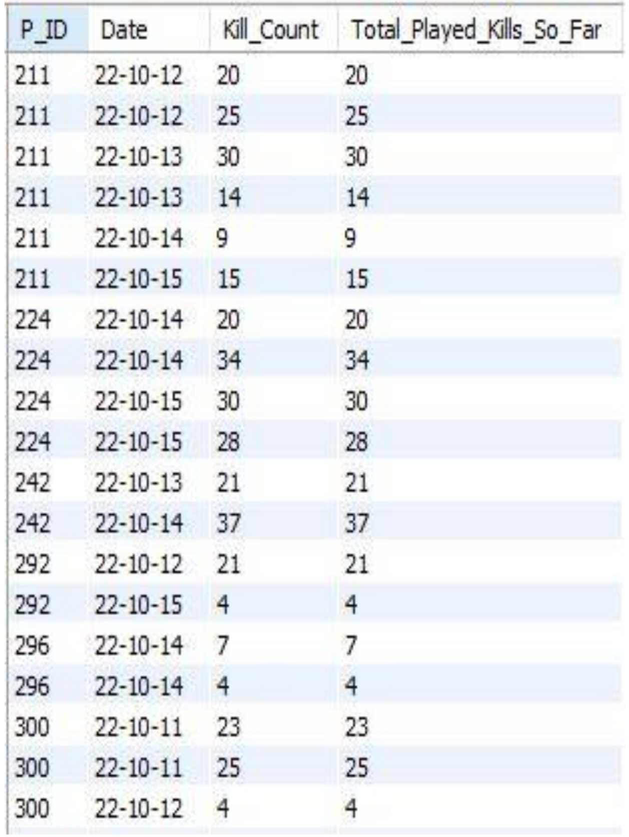
a) Using window functions

```
SELECT P_ID, DATE_FORMAT(start_datetime, '%y-%m-%d’) AS Date, Kill_Count,
SUM(Kill_Count) OVER (PARTITION BY P_ID, DATE_FORMAT(start_datetime, '%y-%m-%d')
ORDER BY start_datetime) AS Total_Played_Kills_So_Far
FROM level_details2
ORDER BY P_ID, start_datetime;
```

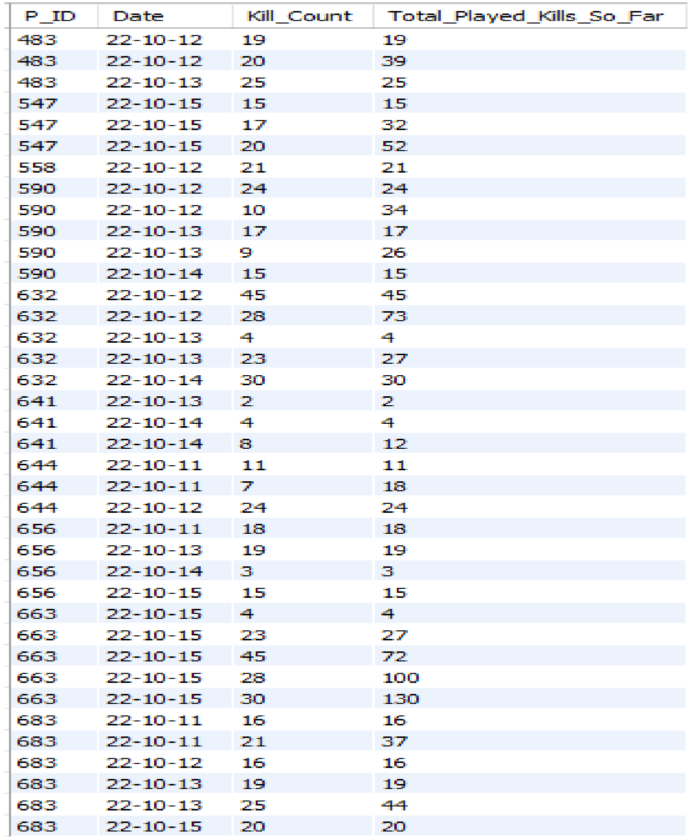
b) Without window functions

```
SELECT P_ID, DATE_FORMAT(start_datetime, '%y-%m-%d’) AS Date, Kill_Count,
SUM(Kill_Count) AS Total_Played_Kills_So_Far
FROM level_details2
GROUP BY P_ID, start_datetime, Kill_Count
ORDER BY P_ID, start_datetime;
```

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



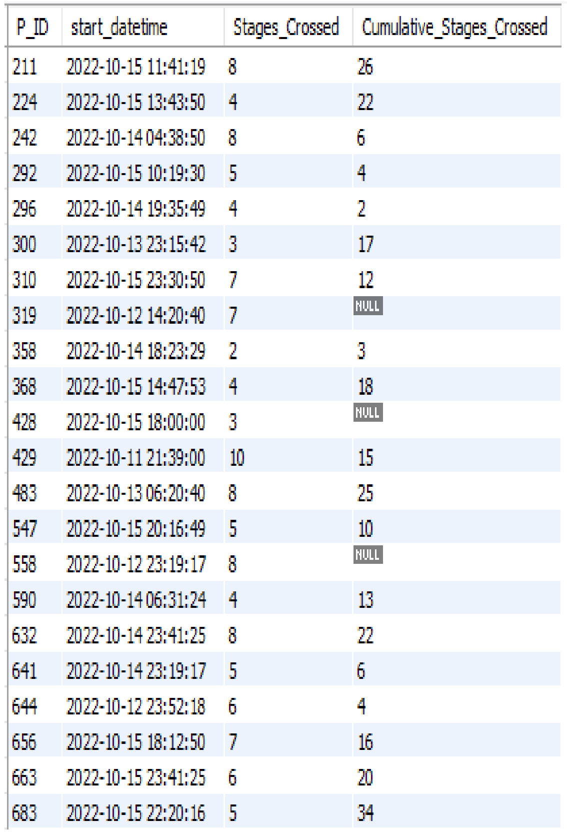
P_ID	Date	Kill_Count	Total_Played_Kills_So_Far
211	22-10-12	20	20
211	22-10-12	25	25
211	22-10-13	30	30
211	22-10-13	14	14
211	22-10-14	9	9
211	22-10-15	15	15
224	22-10-14	20	20
224	22-10-14	34	34
224	22-10-15	30	30
224	22-10-15	28	28
242	22-10-13	21	21
242	22-10-14	37	37
292	22-10-12	21	21
292	22-10-15	4	4
296	22-10-14	7	7
296	22-10-14	4	4
300	22-10-11	23	23
300	22-10-11	25	25
300	22-10-12	4	4



P_ID	Date	Kill_Count	Total_Played_Kills_So_Far
483	22-10-12	19	19
483	22-10-12	20	39
483	22-10-13	25	25
547	22-10-15	15	15
547	22-10-15	17	32
547	22-10-15	20	52
558	22-10-12	21	21
590	22-10-12	24	24
590	22-10-12	10	34
590	22-10-13	17	17
590	22-10-13	9	26
590	22-10-14	15	15
632	22-10-12	45	45
632	22-10-12	28	73
632	22-10-13	4	4
632	22-10-13	23	27
632	22-10-14	30	30
641	22-10-13	2	2
641	22-10-14	4	4
641	22-10-14	8	12
644	22-10-11	11	11
644	22-10-11	7	18
644	22-10-12	24	24
656	22-10-11	18	18
656	22-10-13	19	19
656	22-10-14	3	3
656	22-10-15	15	15
663	22-10-15	4	4
663	22-10-15	23	27
663	22-10-15	45	72
663	22-10-15	28	100
663	22-10-15	30	130
683	22-10-11	16	16
683	22-10-11	21	37
683	22-10-12	16	16
683	22-10-13	19	19
683	22-10-13	25	44
683	22-10-15	20	20

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

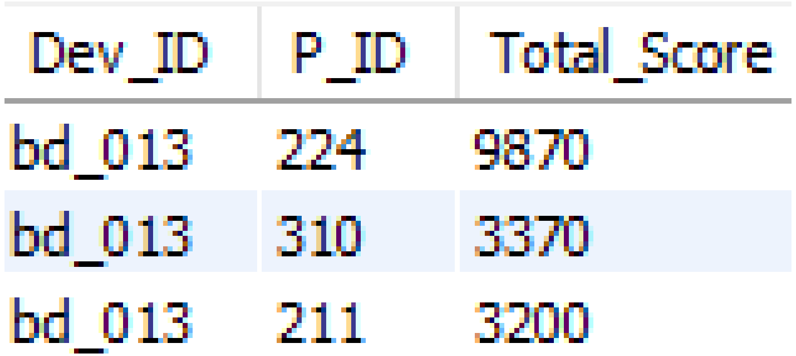
```
SELECT ld.P_ID, ld.start_datetime, ld.Stages_Crossed,
   ( SELECT SUM(Stages_Crossed)
     FROM level_details2
     WHERE P_ID = ld.P_ID
     AND start_datetime < ld.start_datetime )
     AS  Cumulative_Stages_Crossed
FROM level_details2 ld
WHERE NOT EXISTS (
SELECT 1
FROM level_details2 ld2
WHERE ld.P_ID = ld2.P_ID
        AND ld.start_datetime < ld2.start_datetime)
ORDER BY ld.P_ID, ld.start_datetime;
```



P_ID	start_datetime	Stages_Crossed	Cumulative_Stages_Crossed
211	2022-10-15 11:41:19	8	26
224	2022-10-15 13:43:50	4	22
242	2022-10-14 04:38:50	8	6
292	2022-10-15 10:19:30	5	4
296	2022-10-14 19:35:49	4	2
300	2022-10-13 23:15:42	3	17
310	2022-10-15 23:30:50	7	12
319	2022-10-12 14:20:40	7	NULL
358	2022-10-14 18:23:29	2	3
368	2022-10-15 14:47:53	4	18
428	2022-10-15 18:00:00	3	NULL
429	2022-10-11 21:39:00	10	15
483	2022-10-13 06:20:40	8	25
547	2022-10-15 20:16:49	5	10
558	2022-10-12 23:19:17	8	NULL
590	2022-10-14 06:31:24	4	13
632	2022-10-14 23:41:25	8	22
641	2022-10-14 23:19:17	5	6
644	2022-10-12 23:52:18	6	4
656	2022-10-15 18:12:50	7	16
663	2022-10-15 23:41:25	6	20
683	2022-10-15 22:20:16	5	34

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

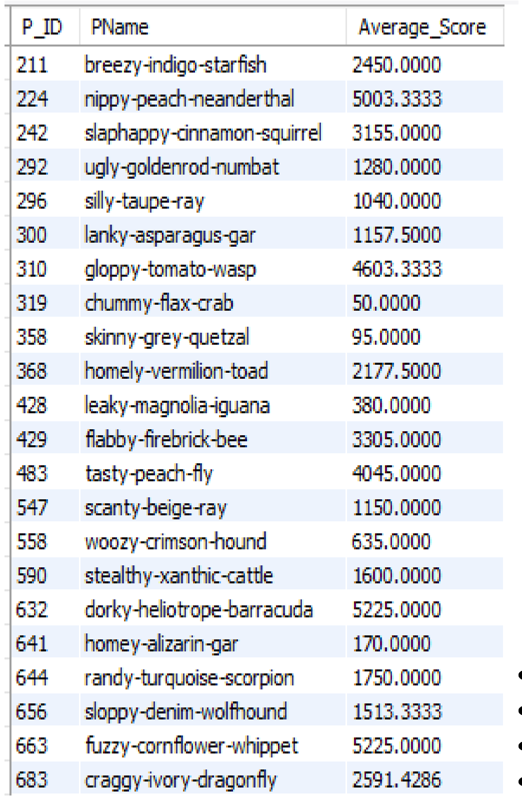
```
SELECT Dev_ID, P_ID, SUM(Score) AS Total_Score
FROM level_details2
GROUP BY Dev_ID, P_ID
ORDER BY Dev_ID, Total_Score DESC
LIMIT 3;
```



Dev_ID	P_ID	Total_Score
bd_013	224	9870
bd_013	310	3370
bd_013	211	3200

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

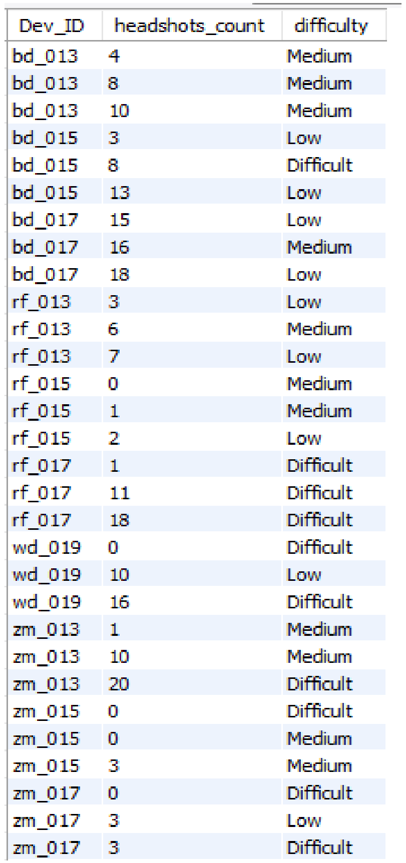
```
SELECT pd.P_ID, pd.PName, AVG(ld2.Score) AS Average_Score
FROM player_details pd
JOIN level_details2 ld2
ON pd.P_ID = ld2.P_ID
JOIN ( SELECT P_ID, AVG(Score) AS Player_Avg_Score
FROM level_details2
GROUP BY P_ID) AS avg_scores
ON pd.P_ID = avg_scores.P_ID
WHERE ld2.Score > 0.5 * avg_scores.Player_Avg_Score
GROUP BY pd.P_ID, pd.PName
```



P_ID	PName	Average_Score
211	breezy-indigo-starfish	2450.0000
224	nippy-peach-neanderthal	5003.3333
242	slaphappy-cinnamon-squirrel	3155.0000
292	ugly-goldenrod-numbat	1280.0000
296	silly-taupe-ray	1040.0000
300	lanky-asparagus-gar	1157.5000
310	gloppy-tomato-wasp	4603.3333
319	chummy-flax-crab	50.0000
358	skinny-grey-quetzal	95.0000
368	homely-vermilion-toad	2177.5000
428	leaky-magnolia-iguana	380.0000
429	flabby-firebrick-bee	3305.0000
483	tasty-peach-fly	4045.0000
547	scanty-beige-ray	1150.0000
558	woozy-crimson-hound	635.0000
590	stealthy-xanthic-cattle	1600.0000
632	dorky-heliotrope-barracuda	5225.0000
641	homey-alizarin-gar	170.0000
644	randy-turquoise-scorpion	1750.0000
656	sloppy-denim-wolfhound	1513.3333
663	fuzzy-cornflower-whippet	5225.0000
683	craggy-ivory-dragonfly	2591.4286

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 FindTopHeadshotsCount(IN n INT)
BEGIN CREATE TEMPORARY TABLE temp_table AS
    SELECT ld.Dev_ID, ld.headshots_count, ld.difficulty,
    ROW_NUMBER()
    OVER (PARTITION BY ld.Dev_ID ORDER BY ld.headshots_count) AS rn
    FROM level_details2 ld
    WHERE ld.headshots_count IS NOT NULL;
    SELECT Dev_ID, headshots_count, difficulty
    FROM temp_table
    WHERE rn <= n;
    DROP TEMPORARY TABLE IF EXISTS temp_table;
END //
DELIMITER ;
CALL FindTopHeadshotsCount(3);
```



Dev_ID	headshots_count	difficulty
bd_013	4	Medium
bd_013	8	Medium
bd_013	10	Medium
bd_015	3	Low
bd_015	8	Difficult
bd_015	13	Low
bd_017	15	Low
bd_017	16	Medium
bd_017	18	Low
rf_013	3	Low
rf_013	6	Medium
rf_013	7	Low
rf_015	0	Medium
rf_015	1	Medium
rf_015	2	Low
rf_017	1	Difficult
rf_017	11	Difficult
rf_017	18	Difficult
wd_019	0	Difficult
wd_019	10	Low
wd_019	16	Difficult
zm_013	1	Medium
zm_013	10	Medium
zm_013	20	Difficult
zm_015	0	Difficult
zm_015	0	Medium
zm_015	3	Medium
zm_017	0	Difficult
zm_017	3	Low
zm_017	3	Difficult

THANK YOU