Gaming Analysis with SQL

Internship Project by Anusiem Adaeze Comfort with Mentorness

BATCH: MIP-DA-06

Table of Contents

About the project

In this project I worked with a dataset related to a game. 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_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

Dataset Description

- Players play a game divided into 3-levels (L0,L1 and L2)
- Each level has 3 difficulty levels (Low,Medium,High)
- At each level,players have to kill the opponents using guns/physical fight
- Each level has multiple stages at each difficulty level.
- A player can only play L1 using its system generated L1_code.
- Only players who have played Level1 can possibly play Level2 using its system generated L2_code.
- By default a player can play L0.
- Each player can login to the game using a Dev_ID.
- Players can earn extra lives at each stage in a level.

Outcomes

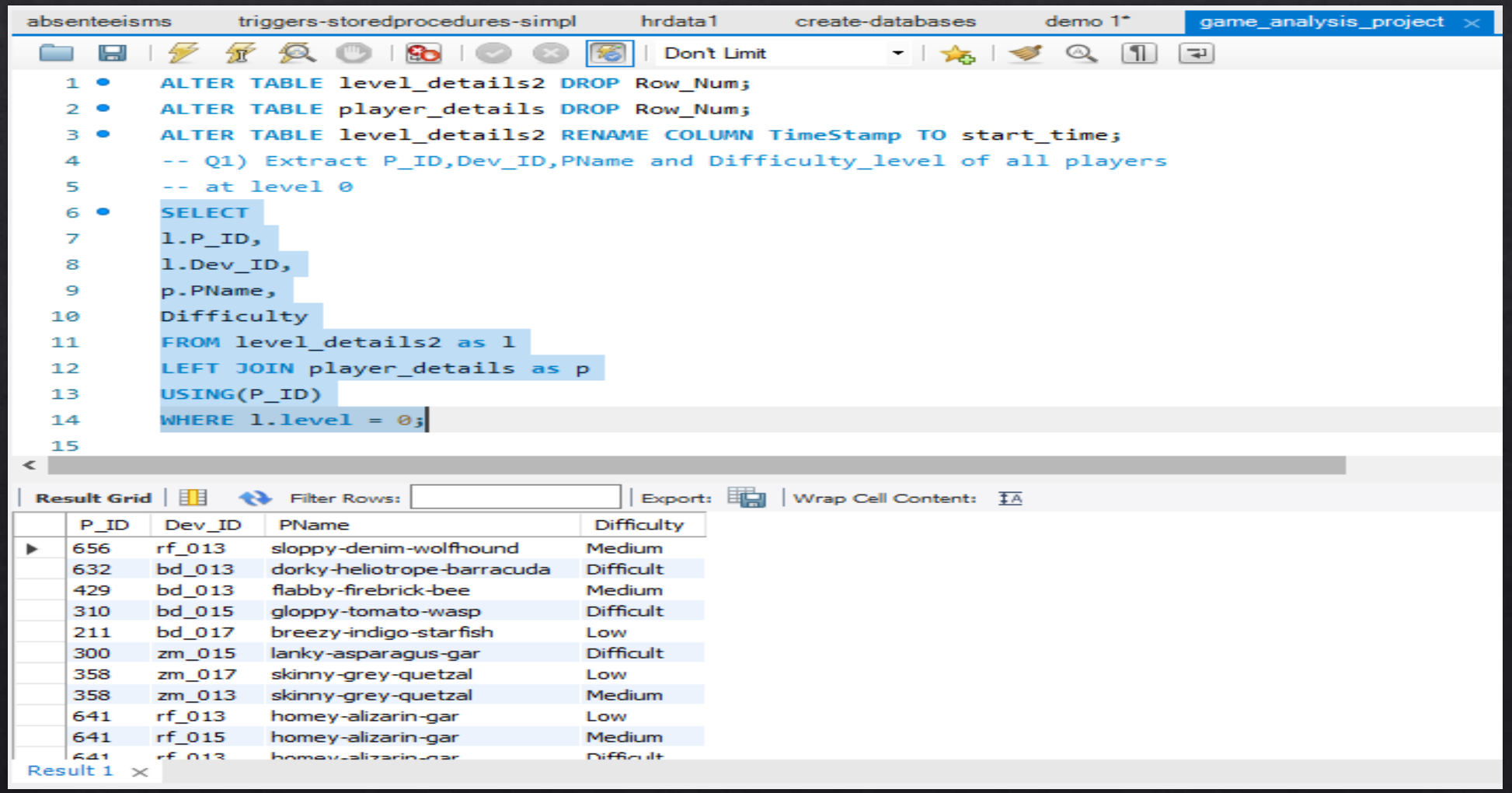
- Gain strong analytical skills
- Enhance critical thinking abilities
- Enhance excellent communication skills
- Sharpen my ability to handle large and big data

New Learnings

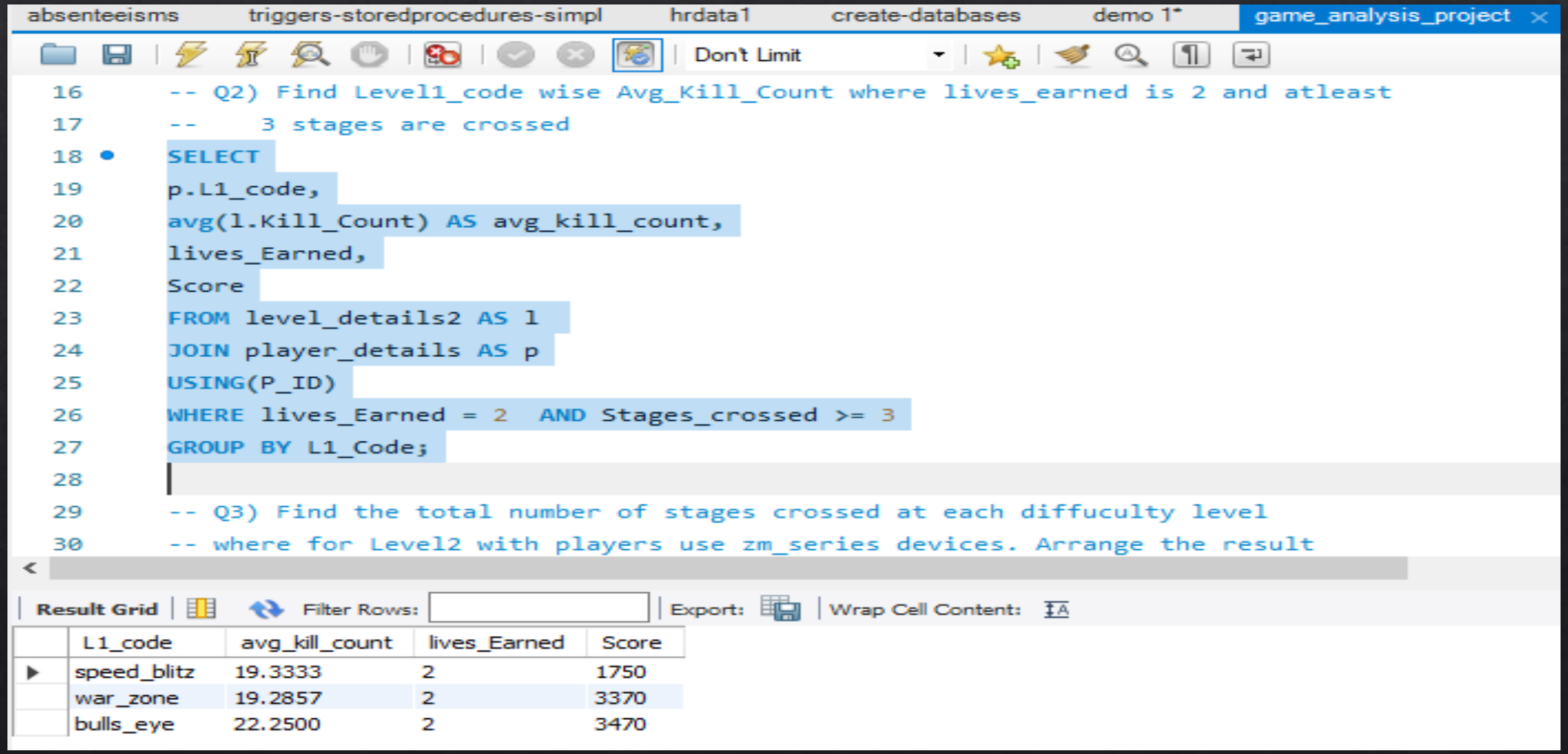
- Stored Procedures are sets of SQL codes that can be created, stored and accessed often
- Ability to use window functions to perform a calculation on an aggregate value based on a set of rows and return multiple rows for each group
- How to use SQL Rank function and Row_Number function to find highest and lowest scores based on categories
- How to use nested queries in SQL

Analyze SQL problem statements

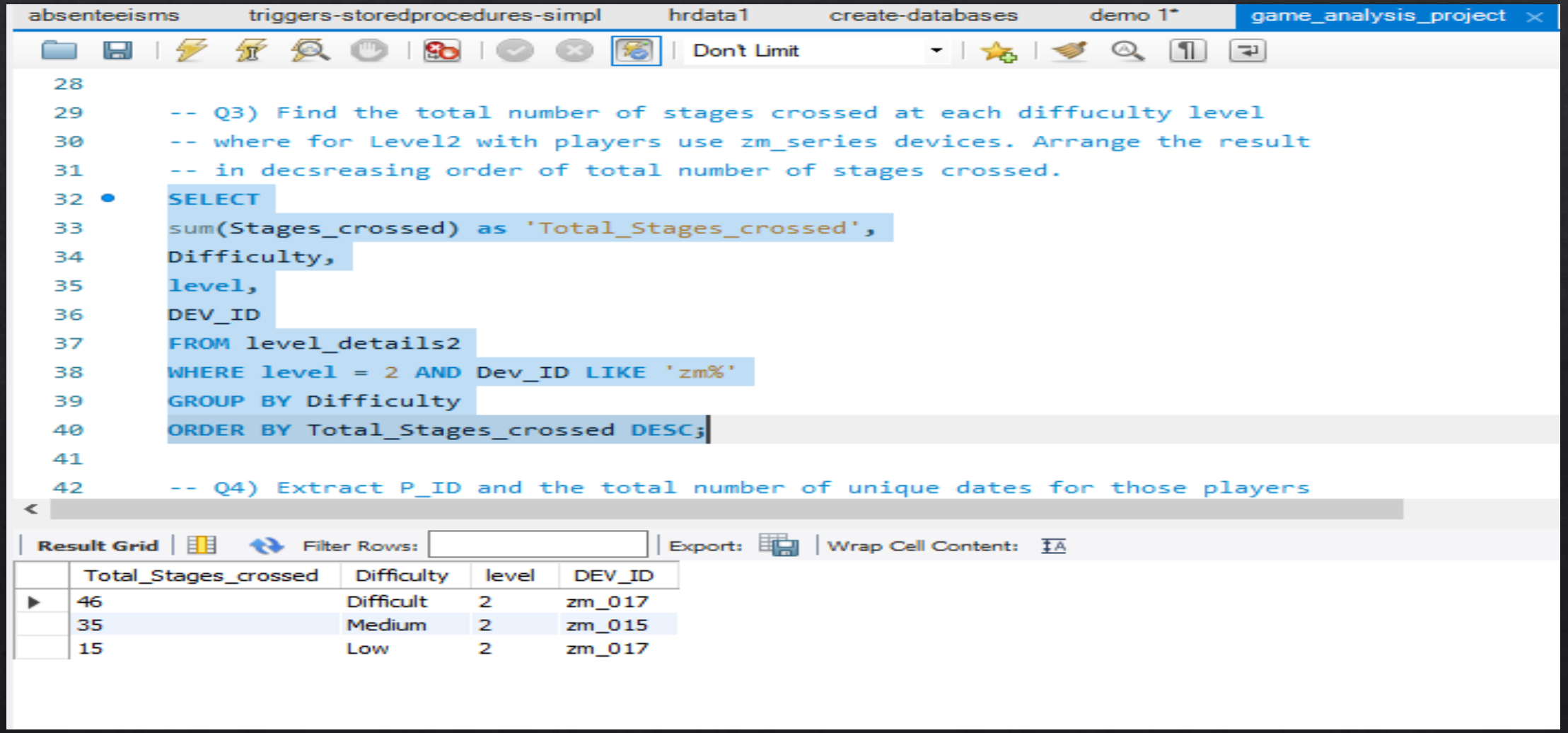
1. Extract `P_ID`, `Dev_ID`, `PName`, and `Difficulty_level` of all players at Level 0.



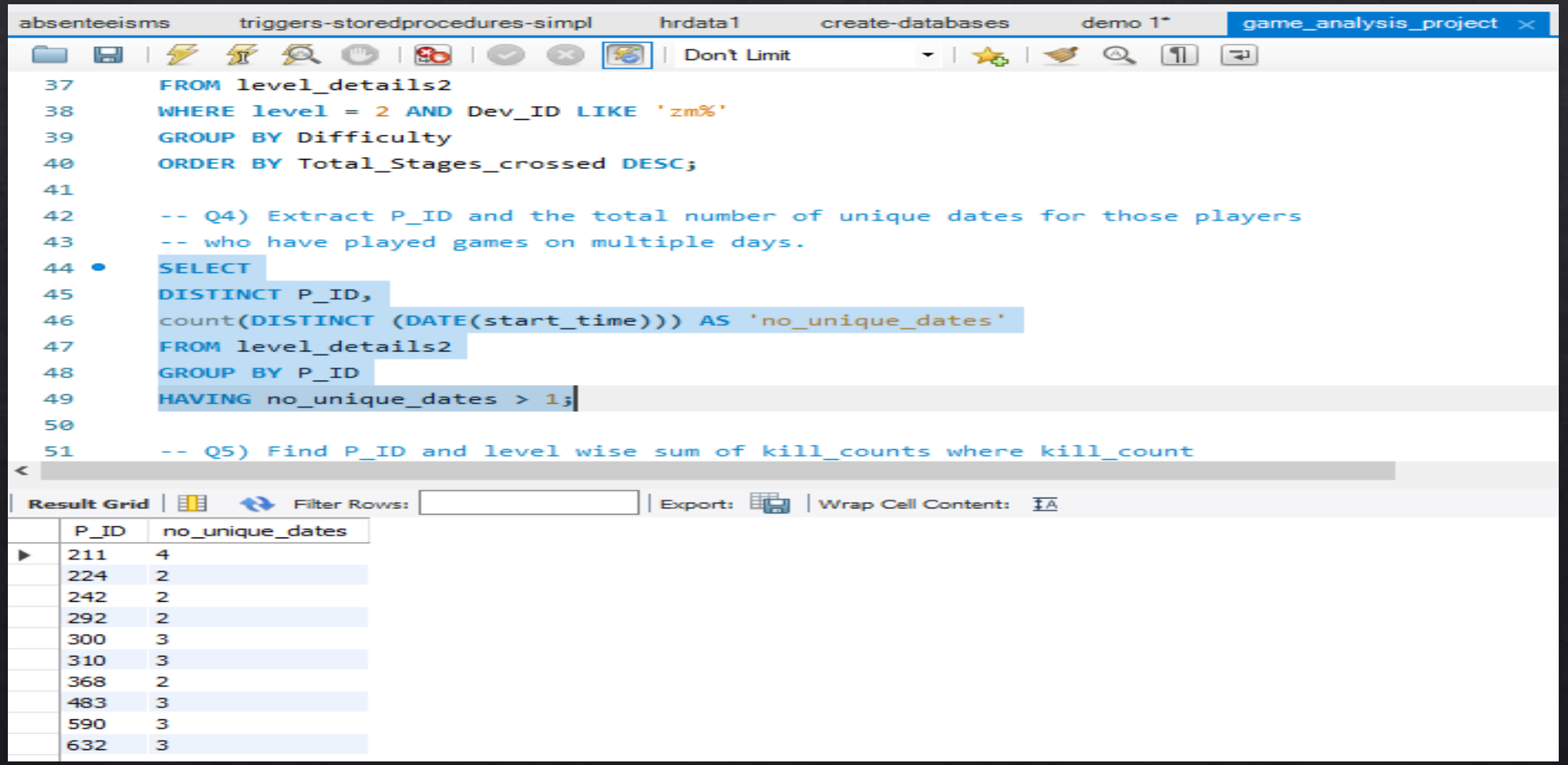
2. Find `Level1_code`wise average `Kill_Count` where `lives_earned` is 2, and at least 3 stages are crossed.



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.

absenteeisms | triggers-storedprocedures-simpl | hrdata1 | create-databases | demo 1* | game_analysis_project | level_details2 | player_details

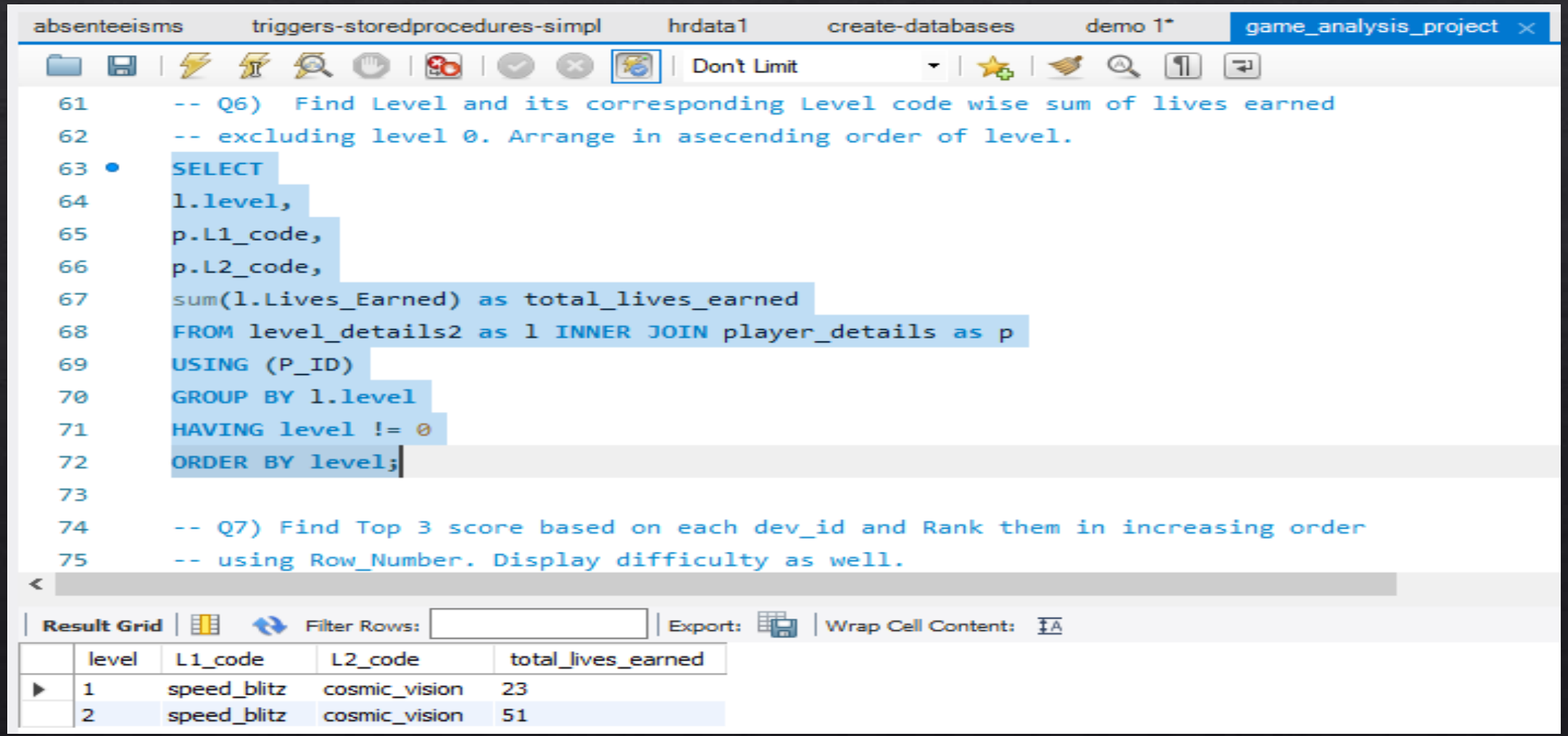
Don't Limit

```
49      HAVING no_unique_dates > 1;
50
51      -- Q5) Find P_ID and level wise sum of kill_counts where kill_count
52      -- is greater than avg kill count for the Medium difficulty.
53      SELECT
54      *,
55      SUM(Kill_Count) as total_kill_count,
56      AVG(Kill_Count) AS 'avg_kill_count'
57      FROM level_details2
58      GROUP BY P_ID, level
59      HAVING Kill_Count > avg_kill_count AND Difficulty = 'Medium';
60
61      -- Q6)  Find Level and its corresponding Level code wise sum of lives earned
62      -- excluding level 0. Arrange in asecending order of level.
63      SELECT
```

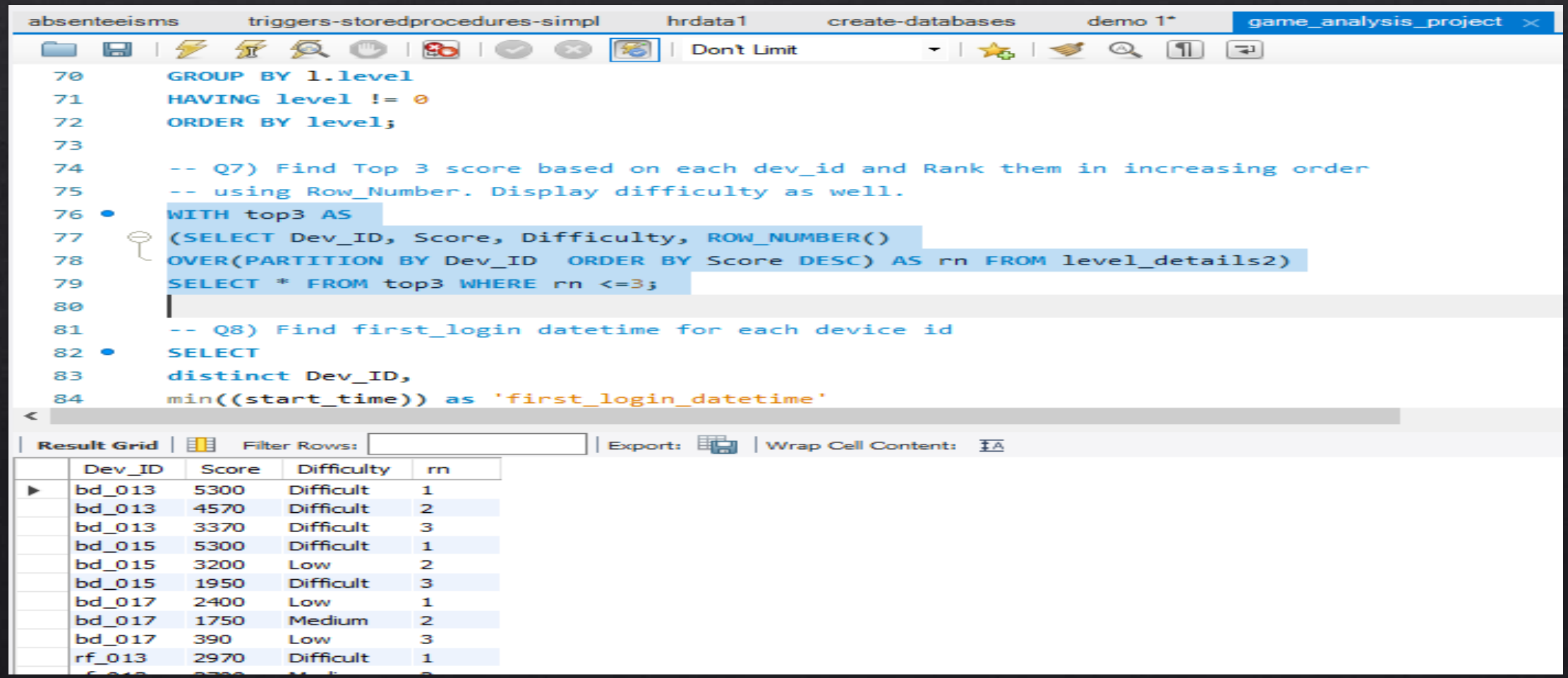
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

P_ID	Dev_ID	start_time	Stages_crossed	Level	Difficulty	Kill_Count	Headshots_Count	Score	Lives_Earned	total_kill_count	avg_kill_count
644	zm_015	2022-10-11 14:05:08	3	1	Medium	11	5	350	1	18	9.0000
483	zm_017	2022-10-11 14:33:27	10	2	Medium	50	43	5490	3	94	31.3333
368	zm_015	2022-10-12 01:14:34	7	1	Medium	20	18	2040	0	34	17.0000
683	rf_015	2022-10-13 22:30:17	5	2	Medium	25	18	2800	0	74	18.5000

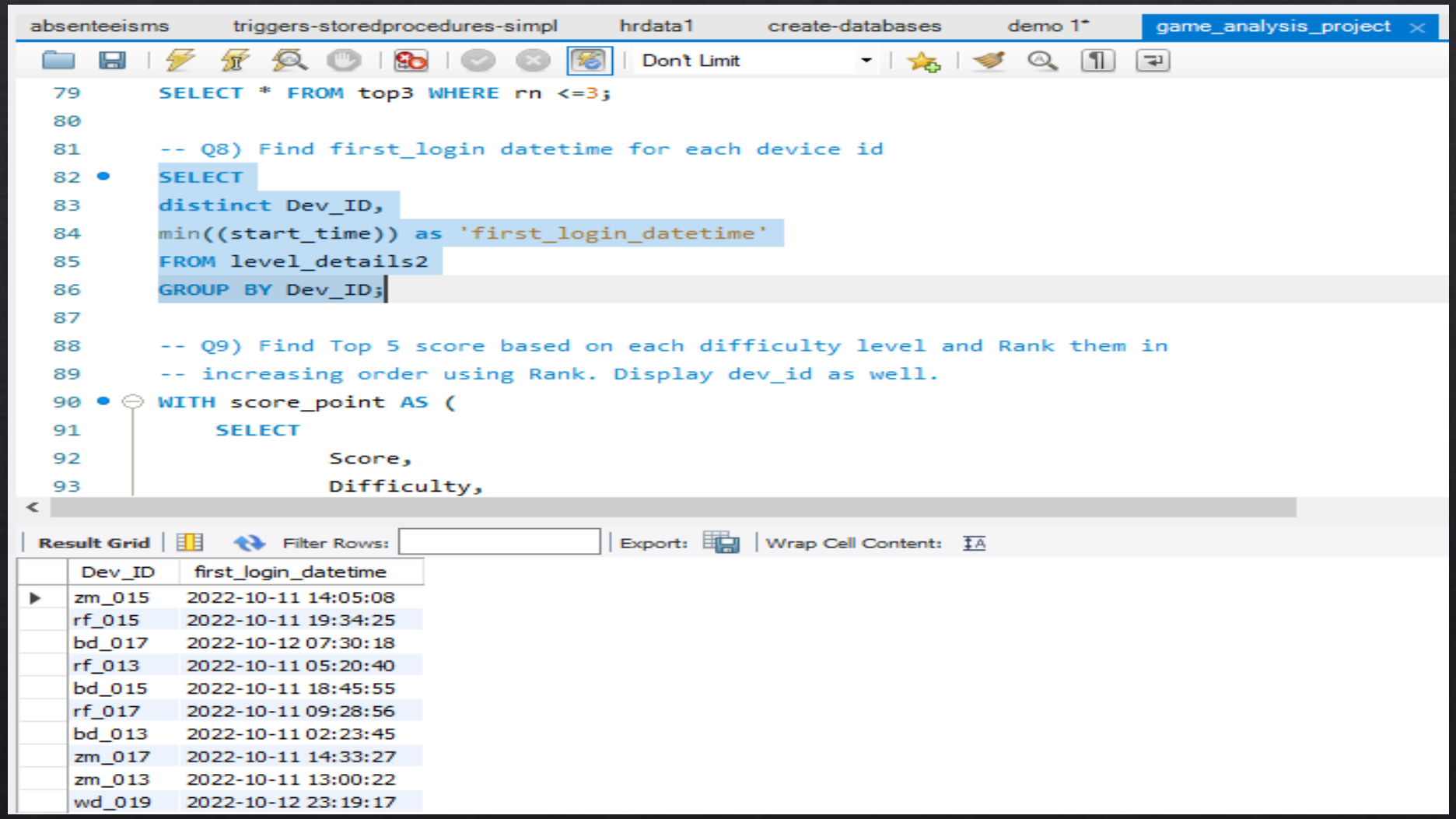
6. Find `Level` and its corresponding `Level_code`wise sum of lives earned, excluding Level 0. Arrange in ascending order of level



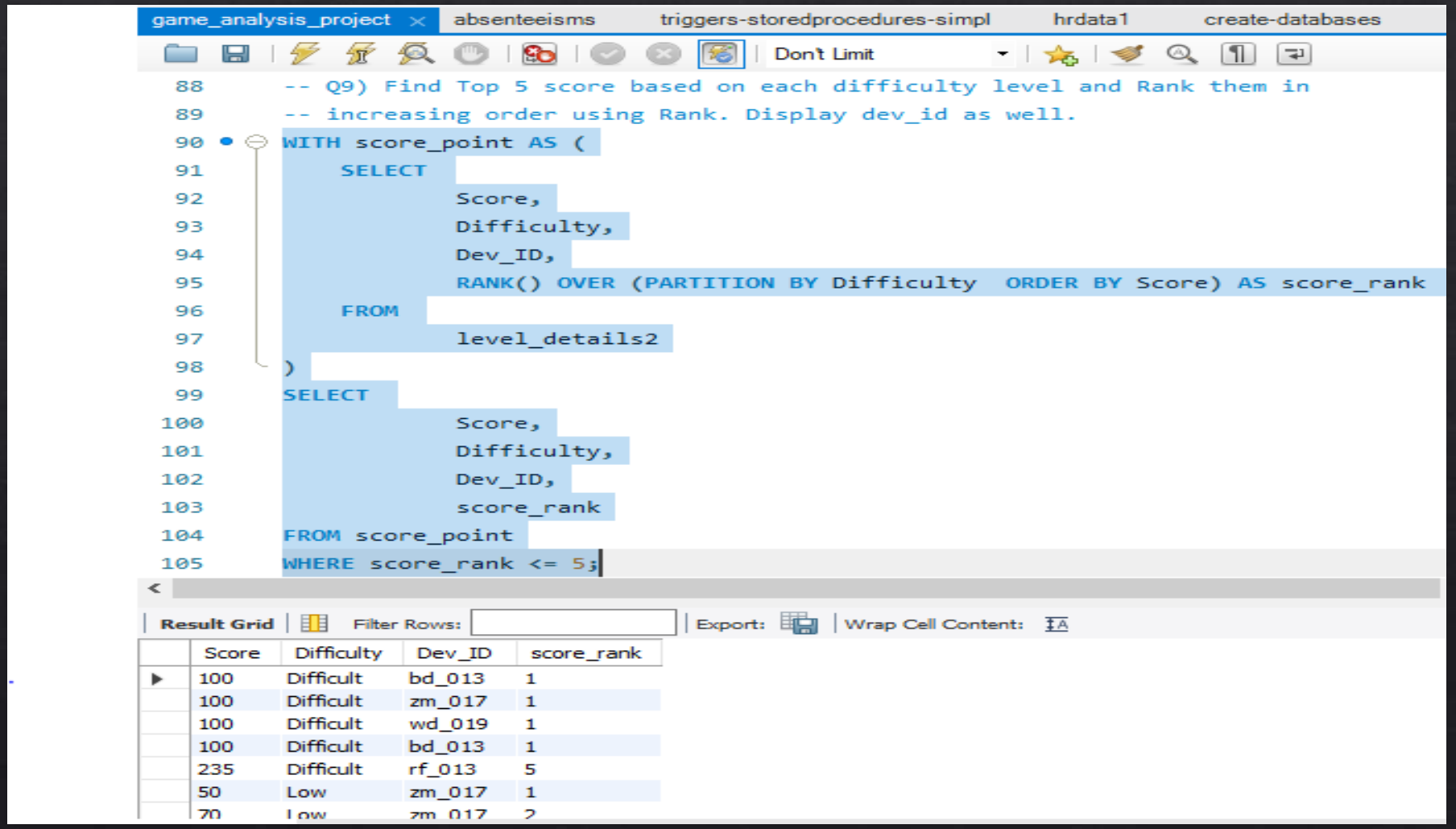
7. Find the top 3 scores based on each `Dev_ID` and rank them in increasing order using `Row_Number`. Display the difficulty as well.



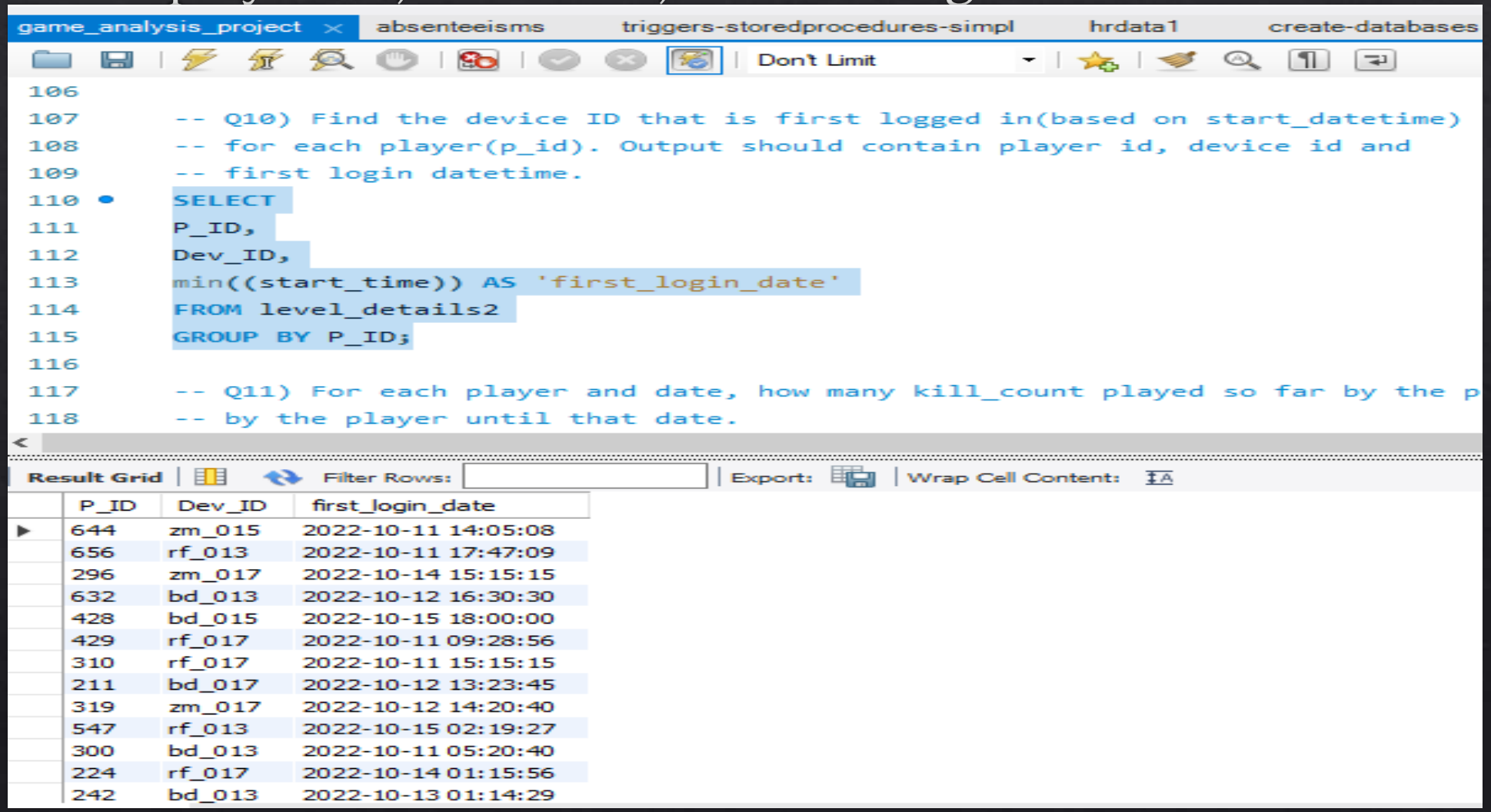
8. Find the `first_login` datetime for each device ID.



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`). Output should contain player ID, device ID, and first login datetime.



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

game_analysis_project | absenteeisms | triggers-storedprocedures-simpl | hrdata1 | create-databases | demo 1* | level_details2 | player_details

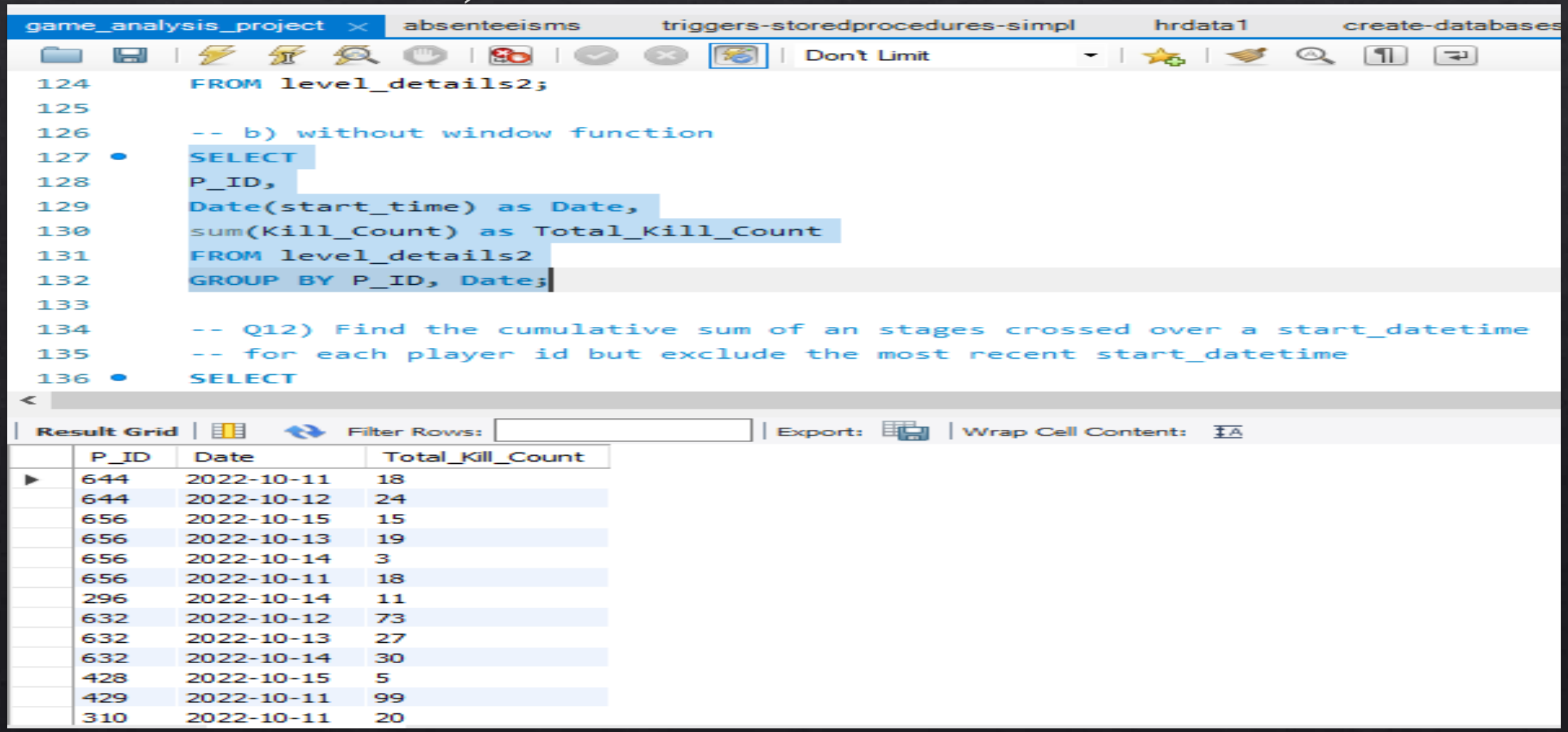
Don't Limit

```
115     GROUP BY P_ID;
116
117     -- Q11) For each player and date, how many kill_count played so far by the player. That is, the total number of games played
118     -- by the player until that date.
119     -- a) window function
120     SELECT
121     P_ID,
122     Date(start_time) as Date,
123     sum(Kill_Count) OVER (PARTITION BY P_ID ORDER BY start_time) as total_number_of_games
124     FROM level_details2;
125
126     -- b) without window function
127     SELECT
```

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

P_ID	Date	total_number_of_games
211	2022-10-12	20
211	2022-10-12	45
211	2022-10-13	75
211	2022-10-13	89
211	2022-10-14	98
211	2022-10-15	113
224	2022-10-14	20
224	2022-10-14	54
224	2022-10-15	84
224	2022-10-15	112
242	2022-10-13	21
242	2022-10-14	58
292	2022-10-12	21

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

game_analysis_project | absenteeisms | triggers-storedprocedures-simpl | hrdata1 | create-databases | demo 1* | level_details2

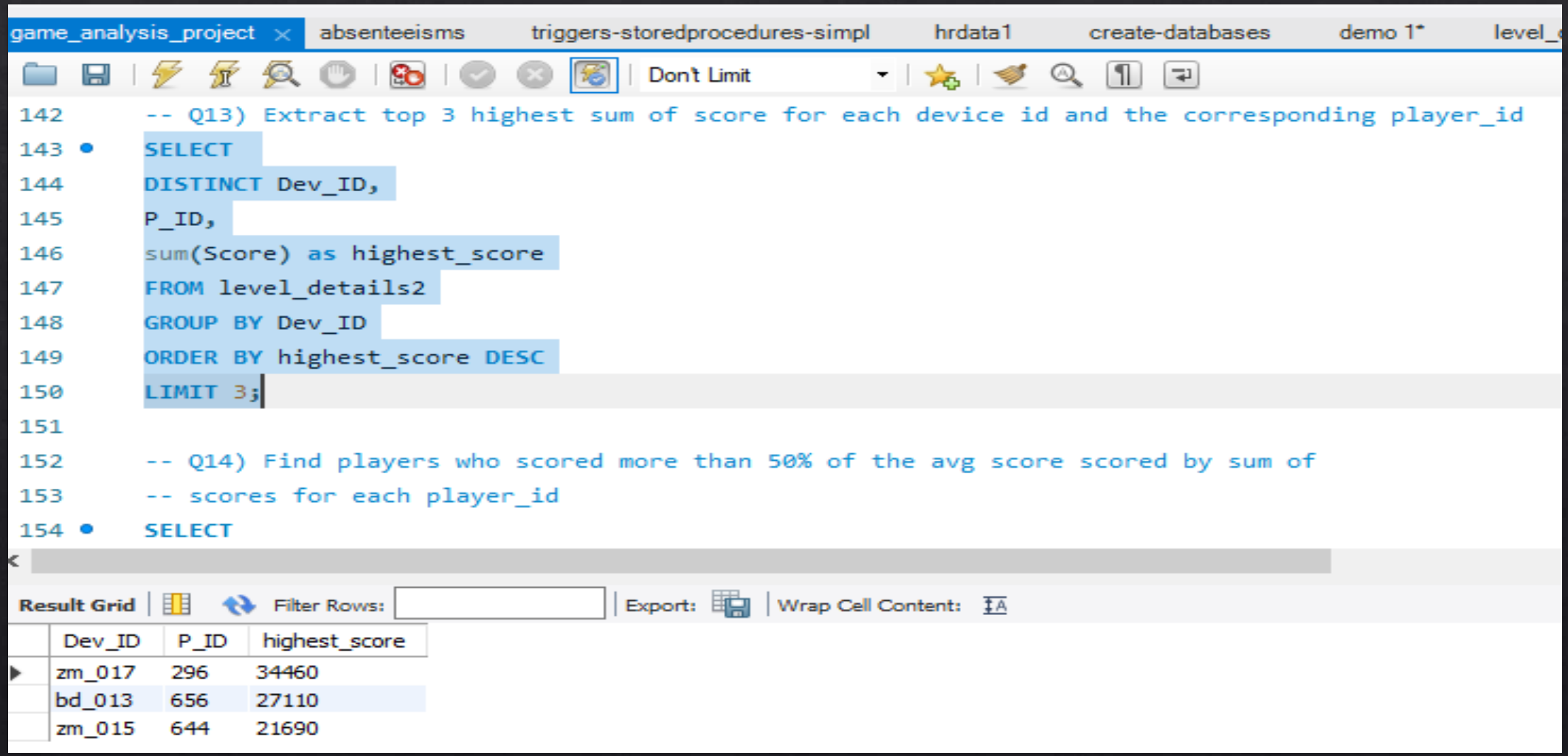
Don't Limit

```
133
134     -- Q12) Find the cumulative sum of an stages crossed over a start_datetime
135     -- for each player id but exclude the most recent start_datetime
136     SELECT
137     P_ID,
138     start_time as Datetime,
139     sum(Stages_crossed) OVER (PARTITION BY P_ID ORDER BY start_time) as  cumulative_sum_of_stages_crossed
140     FROM level_details2;
141
142     -- Q13) Extract top 3 highest sum of score for each device id and the corresponding player_id
143     SELECT
144     DISTINCT Dev_ID,
145     P_ID,
```

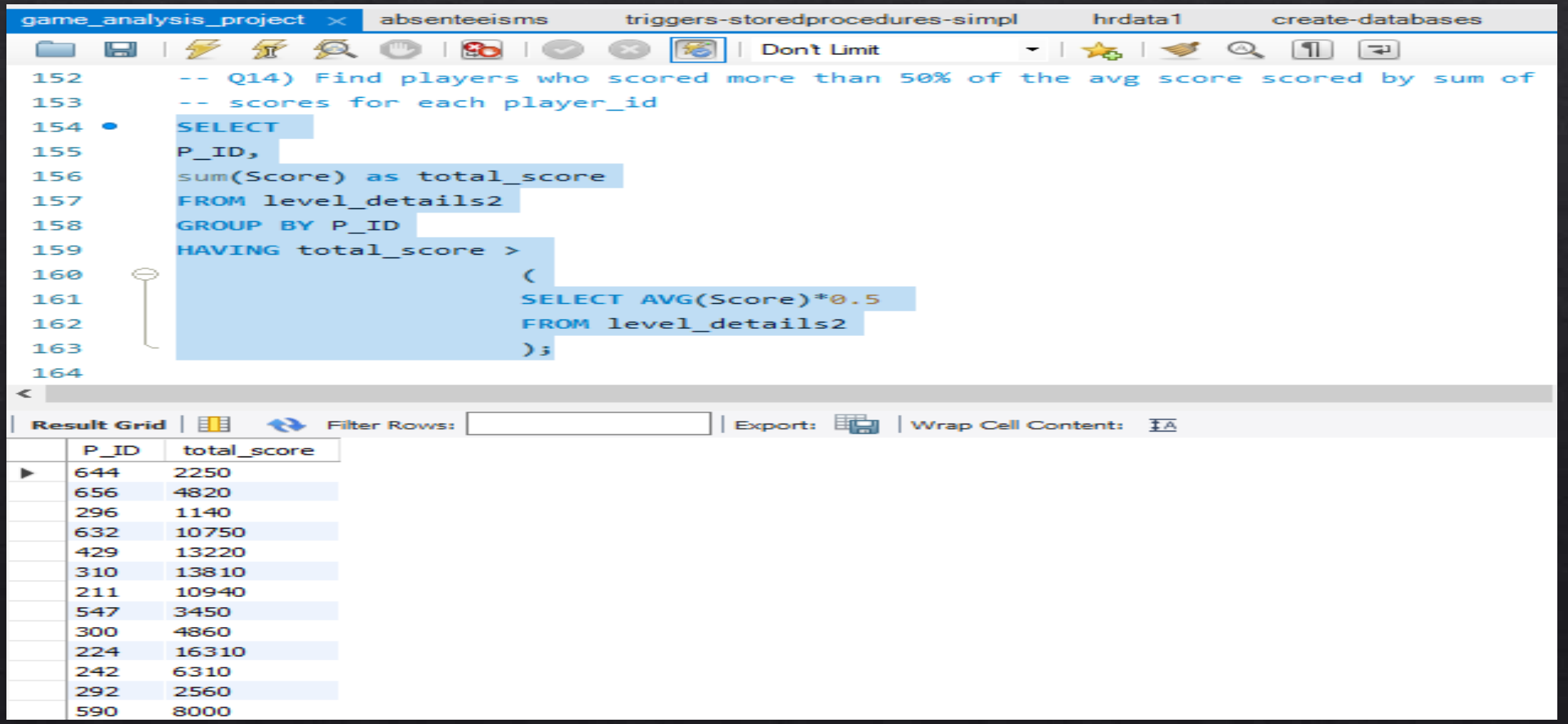
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

P_ID	Datetime	cumulative_sum_of_stages_crossed
211	2022-10-12 13:23:45	4
211	2022-10-12 18:30:30	9
211	2022-10-13 05:36:15	14
211	2022-10-13 22:30:18	19
211	2022-10-14 08:56:24	26
211	2022-10-15 11:41:19	34
224	2022-10-14 01:15:56	7
224	2022-10-14 08:21:49	12
224	2022-10-15 05:30:28	22
224	2022-10-15 13:43:50	26
242	2022-10-13 01:14:29	6
242	2022-10-14 04:38:50	14
292	2022-10-12 04:29:45	4

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

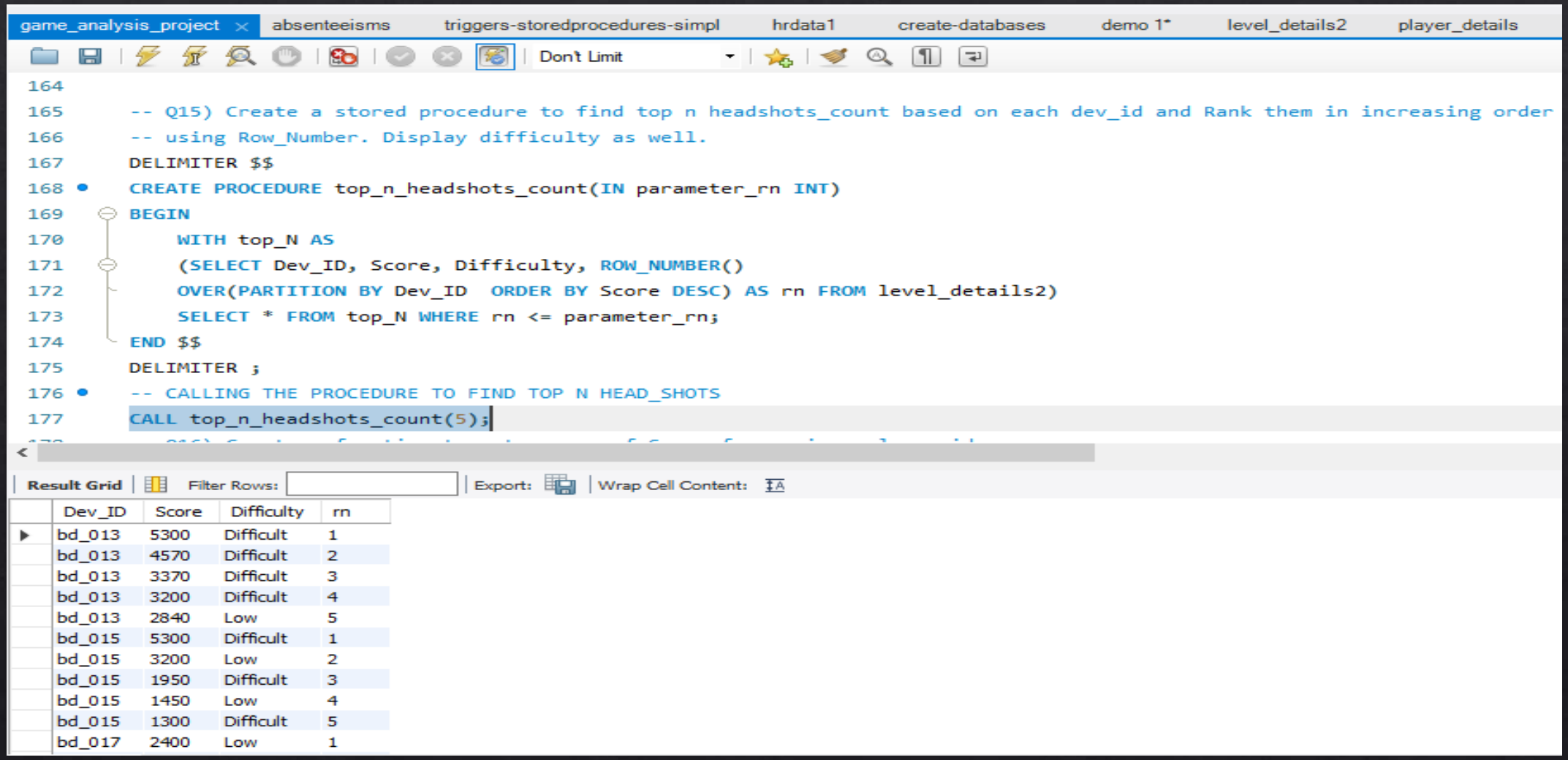


14. Find players who scored more than 50% of the average score, scored by the sum of scores for each `P_ID`.

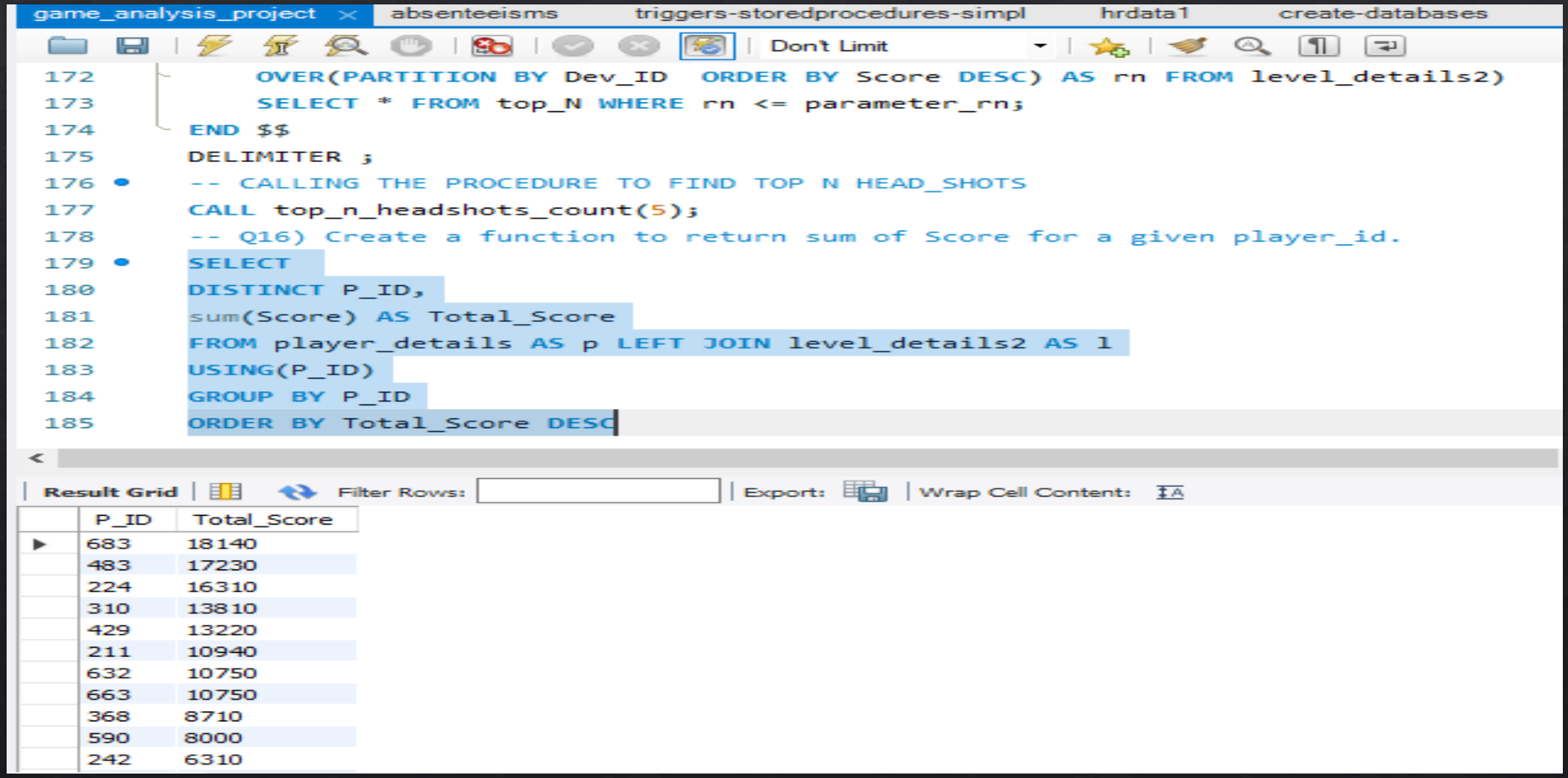


P_ID	total_score
644	2250
656	4820
296	1140
632	10750
429	13220
310	13810
211	10940
547	3450
300	4860
224	16310
242	6310
292	2560
590	8000

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.



17) Create a function to return sum of Score for a given player_id.



Conclusion

SQL is needed to seamlessly navigate through simple and complex datasets, it is used to efficiently execute queries, edit database tables and extract meaningful insights from a dataset to make decisions.

In this game analysis project, using SQL I was able to solve some statement problems, query necessary data to identify pattern and behaviors of the game, to predict gaming outcomes and scores, identify excellent players and most suitable devices they used and many others.