GAME ANALYSIS WITH SQL

" Decoding Gaming Behavior "

BY: Omar Magdy Elmenofy

In this internship, I was working 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 :

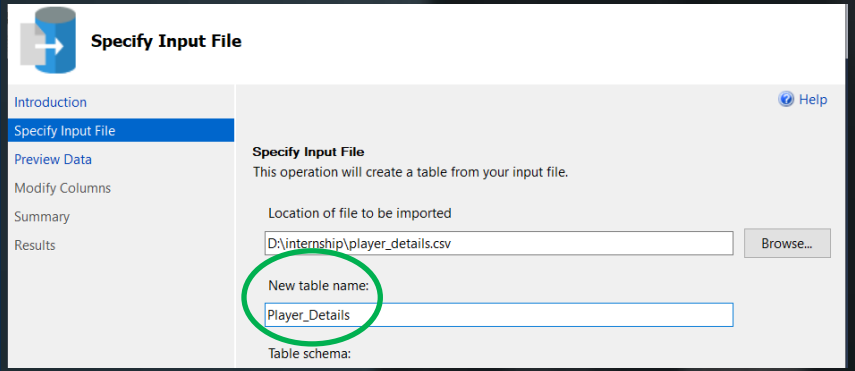
Player Details Table:

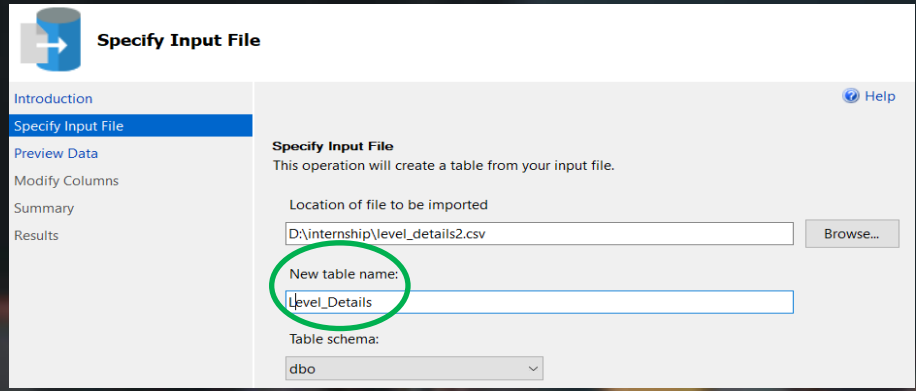
- P_ID → Player ID
- Pname → Player Name
- L1_status → Level 1 Status
- L2_status → Level 2 Status
- L1_code → Systemgenerated Level 1 Code
- L2_code → Systemgenerated Level 2 Code

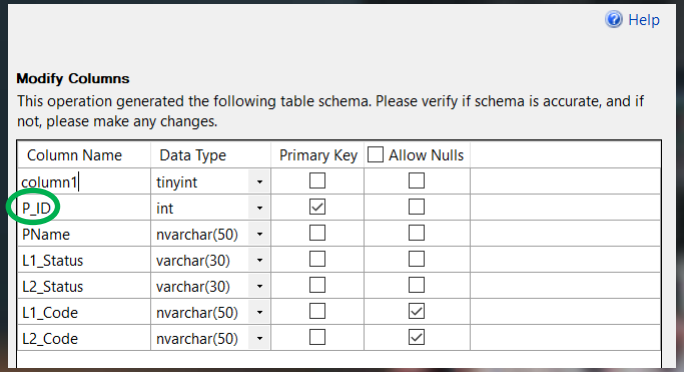
Level Details Table:

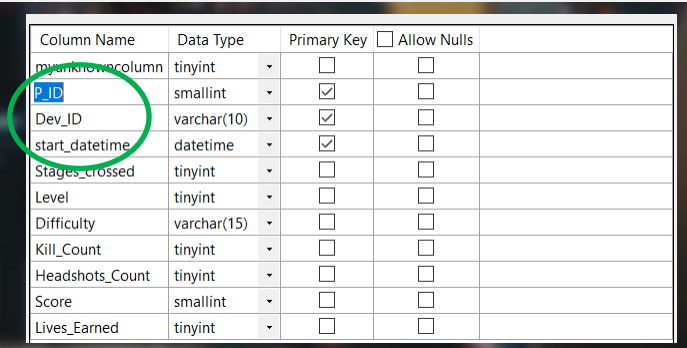
- P_ID → Player ID
- Dev_ID → Device ID
- start_time → Start Time
- stages_crossed → Stages Crossed
- Level → Game Level
- Difficulty → Difficulty Level
- kill_count → Kill Count
- headshots_count → Headshots Count
- Score → Player Score
- lives_earned → Extra Lives Earned

Import the Data









The Source of data is “ CSV “ file

Import the Data

SQLQuery5.sql - (local).Game_Analysis (DESKTOP-LBV1D6M\dell (75))* - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

New Query | Game_Analysis | Execute

Object Explorer

Connect

- Database Snapshots
- Game_Analysis
  - Database Diagrams
  - Tables
    - System Tables
    - FileTables
    - External Tables
    - Graph Tables
    - dbo.Level_Details
      - Columns
        - P_ID (PK, smallint, not null)
        - Dev_ID (PK, varchar(10), not null
        - start_datetime (PK, datetime, no
        - Stages_crossed (tinyint, not null)
        - Level (tinyint, not null)
        - Difficulty (varchar(15), not null)
        - Kill_Count (tinyint, not null)
        - Headshots_Count (tinyint, not nu
        - Score (smallint, not null)
        - Lives_Earned (tinyint, not null)
      - Keys
      - Constraints
      - Triggers
      - Indexes
      - Statistics
    - dbo.Player_Details
      - Columns
        - P_ID (PK, int, not null)
        - PName (nvarchar(50), not null)
        - L1_Status (varchar(30), not null)
        - L2_Status (varchar(30), not null)
        - L1_Code (nvarchar(50), null)
        - L2_Code (nvarchar(50), null)
      - Keys
      - Constraints

SQLQuery5.sql - (l...-LBV1D6M\dell (75))*

```
SELECT *
FROM Player_Details;

SELECT *
FROM Level_Details;
```

100 %

Results | Messages

	P_ID	PName	L1_Status	L2_Status	L1_Code	L2_Code
1	211	breezy-indigo-starfish	1	1	war_zone	splippery_slope
2	224	nippy-peach-neanderthal	1	1	war_zone	splippery_slope
3	242	slaphappy-cinnamon-squirrel	1	0	bulls_eye	NULL
4	292	ugly-goldenrod-numbat	1	0	bulls_eye	NULL
5	296	silly-taupe-ray	1	0	war_zone	NULL
6	300	lanky-asparagus-gar	1	1	speed_blitz	cosmic_vision
7	310	gloppy-tomato-wasp	1	1	war_zone	splippery_slope
8	319	chummy-flax-crab	1	0	speed_blitz	NULL

	P_ID	Dev_ID	start_datetime	Stages_crossed	Level	Difficulty	Kill_Count	Headshots_Count	Score	Lives_Earned
1	211	bd_013	2022-10-12 18:30:50.000	5	1	Difficult	25	15	3200	2
2	211	bd_017	2022-10-12 13:23:45.000	4	0	Low	20	15	390	2
3	211	rf_013	2022-10-13 05:36:15.000	5	1	Medium	30	11	2700	1
4	211	rf_017	2022-10-15 11:41:19.000	8	2	Difficult	15	11	1100	1
5	211	zm_0...	2022-10-13 22:30:18.000	5	2	Low	14	8	2800	0

Query executed successfully. | (local) (16.0 RTM) | DESKTOP-LBV1D6M\dell (75) | Game_Analysis | 00:00:00 | 107 rows

Game ANALYSIS WITH SQL

SQLQuery3.sql - (l...-LBV1D6M\dell (63))*

```sql
SELECT P.P_ID, L.Dev_ID, P.PNAME, L.difficulty Difficulty_level
FROM Player_Details P
JOIN Level_Details L ON P.P_ID = L.P_ID
WHERE L.level= 0;
```

100 %

Results | Messages

	P_ID	Dev_ID	PNAME	Difficulty_level
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_013	skinny-grey-quetzal	Medium
5	358	zm_017	skinny-grey-quetzal	Low
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	homey-alizarin-gar	Low
10	641	rf_013	homey-alizarin-gar	Difficult
11	641	rf_015	homey-alizarin-gar	Medium
12	656	rf_013	sloppy-denim-wolfhound	Medium

Query executed successfully.

Q 01

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

JOIN

Game ANALYSIS WITH SQL

SQLQuery3.sql - (l...-LBV1D6M\dell (63))*

```sql
SELECT P.L1_code, Avg (L.kill_count) as avg_kill_count
FROM Player_Details P
JOIN Level_Details L ON P.P_ID = L.P_ID
WHERE L.lives_earned = 2 AND L.stages_crossed >= 3
GROUP BY P.L1_code;
```

100 %

Results | Messages

	L1_code	avg_kill_count
1	bulls_eye	22
2	speed_blitz	19
3	war_zone	19

Query executed successfully.

Q 02

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

JOIN

Aggregation Function

Game ANALYSIS WITH SQL

SQLQuery4.sql - (l...-LBV1D6M\dell (76)) | SQLQuery3.sql - (l...-LBV1D6M\dell (63))*

```sql
SELECT L.Difficulty, SUM(L.Stages_crossed) AS total_stages_crossed
FROM Level_Details L
JOIN Player_Details P ON L.P_ID = P.P_ID
WHERE  L.Dev_ID LIKE 'zm%' AND L.Level = 2
GROUP BY L.difficulty
ORDER BY total_stages_crossed DESC;
```

Q 03

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.

LIKE

100 %

Results | Messages

	Difficulty	total_stages_crossed
1	Difficult	46
2	Medium	35
3	Low	15

Query executed successfully.

Game ANALYSIS WITH SQL

SQLQuery4.sql - (l...-LBV1D6M\dell (76)) | SQLQuery3.sql - (l...-LBV1D6M\dell (63))*

```sql
SELECT P_ID, COUNT (DISTINCT(start_datetime)) as total_unique_dates
FROM Level_Details
GROUP BY P_ID
HAVING COUNT (DISTINCT(start_datetime)) > 1;
```

Q 04

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

Having

100 %

Results | Messages

	P_ID	total_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

Query executed successfully.

Game ANALYSIS WITH SQL

SQLQuery5.sql - (l...-LBV1D6M\dell (72)) | SQLQuery3.sql - (l...-LBV1D6M\dell (63))*

```sql
SELECT P_ID, level, SUM (kill_count) AS SUM_KILL_COUNT
FROM Level_Details
WHERE kill_count > (select AVG (kill_count) FROM Level_Details WHERE Difficulty = 'Medium' )
GROUP BY P_ID, level;
```

100 %

Results | Messages

	P_ID	level	SUM_KILL_COUNT
1	211	0	20
2	310	0	34
3	558	0	21
4	632	0	45
5	211	1	55
6	224	1	54
7	242	1	58
8	292	1	21
9	300	1	48
10	310	1	20
11	368	1	20
12	429	1	30
13	483	1	40
14	547	1	20

Query executed successfully.

Q 05

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

Subquery

Game ANALYSIS WITH SQL

SQLQuery5.sql - (l...-LBV1D6M\dell (72)) | SQLQuery3.sql - (l...-LBV1D6M\dell (63))*

```sql
SELECT L.level, P.L1_code, P.L2_Code, SUM (L.lives_earned) AS total_lives_earned
FROM Level_Details L
JOIN Player_Details P ON L.P_ID = P.P_ID
WHERE L.level NOT IN (0)
GROUP BY L.level, P.L1_code, P.L2_Code
ORDER BY L.level ASC;
```

Q 06

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

Join, NOT IN

100 %

Results | Messages

	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_faith	NULL	0
5	1	speed_blitz	NULL	0
6	1	speed_blitz	cosmic_vision	4
7	1	speed_blitz	splippery_slope	3
8	1	war_zone	NULL	4
9	1	war_zone	resurgence	0
10	1	war_zone	splippery_slope	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_slope	14

Query executed successfully.

Game ANALYSIS WITH SQL

SQLQuery1.sql - (l...-LBV1D6M\dell (63))*

```sql
SELECT TOP 3 (score), Dev_ID, Difficulty, ROW_NUMBER() OVER (PARTITION BY Dev_ID ORDER BY score DESC) AS ROWNUMBER
FROM Level_Details
GROUP BY Dev_ID, Score,Difficulty;
```

100 %

Results | Messages

	score	Dev_ID	Difficulty	ROWNUMBER
1	5300	bd_013	Difficult	1
2	4570	bd_013	Difficult	2
3	3370	bd_013	Difficult	3

Query executed successfully.

Q 07

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

TOP, ROW_NUMBER() Function

Game ANALYSIS WITH SQL

SQLQuery5.sql - (l...-LBV1D6M\dell (72)) | SQLQuery3.sql - (l...-LBV1D6M\dell (63))*

```sql
SELECT DEV_ID , MIN (start_datetime) AS MIN_Start_datetime
FROM Level_Details
GROUP BY DEV_ID;
```

100 %

Results | Messages

	DEV_ID	MIN_Start_datetime
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

Query executed successfully.

Q 08

Find the `first_login` datetime for each device ID

MIN ()

Game ANALYSIS WITH SQL

SQLQuery5.sql - (l...-LBV1D6M\dell (72)) | SQLQuery3.sql - (l...-LBV1D6M\dell (63))*

```sql
WITH Ranked_difficulty AS (
        SELECT
             DEV_ID, score, difficulty, Rank () over ( partition by difficulty order by score DESC ) as Ranked
        FROM Level_Details)
SELECT Dev_ID, Difficulty, Score, Ranked
FROM Ranked_difficulty
WHERE Ranked < 6;
```

100 %

Results | Messages

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

Query executed successfully.

Q 09

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

CTE, Rank () Function

Game ANALYSIS WITH SQL

SQLQuery5.sql - not connected | SQLQuery3.sql - (l...-LBV1D6M\dell (61))*

```sql
SELECT DEV_ID,P_ID, MIN (start_datetime) AS MIN_start_datetime
FROM Level_Details
GROUP BY Dev_ID,P_ID
```

100 %

Results | Messages

	DEV_ID	P_ID	MIN_start_datetime
1	bd_013	211	2022-10-12 18:30:30.000
2	bd_017	211	2022-10-12 13:23:45.000
3	rf_013	211	2022-10-13 05:36:15.000
4	rf_017	211	2022-10-15 11:41:19.000
5	zm_015	211	2022-10-13 22:30:18.000
6	zm_017	211	2022-10-14 08:56:24.000
7	bd_013	224	2022-10-15 05:30:28.000
8	bd_015	224	2022-10-14 08:21:49.000
9	rf_017	224	2022-10-14 01:15:56.000
10	bd_013	242	2022-10-13 01:14:29.000
11	zm_015	242	2022-10-14 04:38:50.000
12	rf_013	292	2022-10-12 04:29:45.000
13	rf_015	292	2022-10-15 10:19:30.000
14	zm_015	296	2022-10-14 19:35:49.000
15	zm_017	296	2022-10-14 15:15:15.000
16	bd_013	300	2022-10-11 19:19:19.000
17	rf_013	300	2022-10-11 05:20:40.000
18	zm_015	300	2022-10-12 01:45:17.000
19	bd_013	310	2022-10-15 23:30:50.000
20	bd_015	310	2022-10-13 19:18:20.000
21	rf_017	310	2022-10-11 15:15:15.000

Query executed successfully.

Q 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.

MIN ()
→ Q08

Game ANALYSIS WITH SQL

WITH WINDOW FUNCTION

SQLQuery5.sql - not connected | SQLQuery3.sql - (l...-LBV1D6M\dell (61))*

```sql
---WITH WINDOW FUNCTION
SELECT
    P_ID,
    start_datetime,
    SUM(Kill_Count) OVER (PARTITION BY P_ID, start_datetime ORDER BY start_datetime) AS total_kills
FROM
    Level_Details;
```

100 %

Results | Messages

	P_ID	start_datetime	total_kills
1	211	2022-10-12 13:23:45.000	20
2	211	2022-10-12 18:30:30.000	25
3	211	2022-10-13 05:36:15.000	30
4	211	2022-10-13 22:30:18.000	14
5	211	2022-10-14 08:56:24.000	9
6	211	2022-10-15 11:41:19.000	15
7	224	2022-10-14 01:15:56.000	20
8	224	2022-10-14 08:21:49.000	34
9	224	2022-10-15 05:30:28.000	30
10	224	2022-10-15 13:43:50.000	28
11	242	2022-10-13 01:14:29.000	21
12	242	2022-10-14 04:38:50.000	37
13	292	2022-10-12 04:29:45.000	21
14	292	2022-10-15 10:19:30.000	4
15	296	2022-10-14 15:15:15.000	7
16	296	2022-10-14 19:35:49.000	4
17	300	2022-10-11 05:20:40.000	23
18	300	2022-10-11 19:19:19.000	25
19	300	2022-10-12 01:45:17.000	4
20	300	2022-10-12 11:21:20.000	14

Query executed successfully.

Q 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

SQLQuery5.sql - not connected | SQLQuery3.sql - (l...-LBV1D6M\dell (61))*

```sql
--- WITHOUT WINDOW FUNCTION
SELECT
    P_ID,
    start_datetime,
    SUM(Kill_Count) AS SUM_Kill_Count
FROM
    Level_Details
GROUP BY P_ID, start_datetime
ORDER BY P_ID,start_datetime;
```

100 %

Results | Messages

	P_ID	start_datetime	SUM_Kill_Count
1	211	2022-10-12 13:23:45.000	20
2	211	2022-10-12 18:30:30.000	25
3	211	2022-10-13 05:36:15.000	30
4	211	2022-10-13 22:30:18.000	14
5	211	2022-10-14 08:56:24.000	9
6	211	2022-10-15 11:41:19.000	15
7	224	2022-10-14 01:15:56.000	20
8	224	2022-10-14 08:21:49.000	34
9	224	2022-10-15 05:30:28.000	30
10	224	2022-10-15 13:43:50.000	28
11	242	2022-10-13 01:14:29.000	21
12	242	2022-10-14 04:38:50.000	37
13	292	2022-10-12 04:29:45.000	21
14	292	2022-10-15 10:19:30.000	4
15	296	2022-10-14 15:15:15.000	7
16	296	2022-10-14 19:35:49.000	4
17	300	2022-10-11 05:20:40.000	23
18	300	2022-10-11 19:19:19.000	25

Query executed successfully.

WITHOUT WINDOW FUNCTION

Game ANALYSIS WITH SQL

Query1.sql - (l...-LBV1D6M\dell (63))*

```sql
WITH NEW_TABLE AS (
    SELECT
        P_ID,
        Stages_crossed,
        start_datetime,
        ROW_NUMBER() OVER (PARTITION BY P_ID ORDER BY start_datetime DESC) AS RN
    FROM
        Level_Details
)

SELECT
    P_ID, start_datetime,
    SUM(Stages_crossed) AS cumulative_stages_crossed
FROM
    NEW_TABLE
WHERE
    RN > 1
GROUP BY
    P_ID, start_datetime;
```

100 %

Results | Messages

	P_ID	start_datetime	cumulative_stages_crossed
1	683	2022-10-11 02:23:45.000	4
2	300	2022-10-11 05:20:40.000	7
3	429	2022-10-11 09:28:56.000	2
4	429	2022-10-11 13:00:22.000	7
5	644	2022-10-11 14:05:08.000	3
6	483	2022-10-11 14:33:27.000	10
7	310	2022-10-11 15:15:15.000	7
8	656	2022-10-11 17:47:09.000	10
9	683	2022-10-11 18:45:55.000	3
10	300	2022-10-11 19:19:19.000	5
11	429	2022-10-11 19:28:43.000	6
12	644	2022-10-11 19:34:25.000	1
13	483	2022-10-11 22:20:10.000	5
14	368	2022-10-12 01:14:34.000	7
15	300	2022-10-12 01:45:17.000	2
16	483	2022-10-12 02:40:20.000	7
17	368	2022-10-12 04:20:30.000	5
18	292	2022-10-12 04:29:45.000	4

Query executed successfully.

Q 12

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

CTE, ROW_NUMBER() Function

Game ANALYSIS WITH SQL

SQLQuery1.sql - (l...-LBV1D6M\dell (63))*

```sql
SELECT TOP 3 SUM(score) AS Highest_score, Dev_ID, P_ID
FROM Level_Details
GROUP BY Dev_ID,P_ID
ORDER BY Highest_score DESC;
```

100 %

Results | Messages

	Highest_score	Dev_ID	P_ID
1	9870	bd_013	224
2	8900	zm_017	683
3	5600	zm_017	632

Query executed successfully.

Q 13

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

Game ANALYSIS WITH SQL

SQLQuery5.sql - not connected | SQLQuery3.sql - (l...-LBV1D6M\dell (61))*

```sql
SELECT P_ID, SUM (Score) AS SUM_Score
FROM Level_Details
GROUP BY P_ID
HAVING SUM(score) > 0.5 * (SELECT AVG(Score) FROM Level_Details)
```

Results | Messages

	P_ID	SUM_Score
1	211	10940
2	224	16310
3	242	6310
4	292	2560
5	296	1140
6	300	4860
7	310	13810
8	368	8710
9	429	13220
10	483	17230
11	547	3450
12	590	8000
13	632	10750
14	644	2250
15	656	4820
16	663	10750
17	683	18140

Query executed successfully.

Q 14

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

Having

Game ANALYSIS WITH SQL

stored procedure

Q 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.

SQLQuery3.sql - (l...-LBV1D6M\dell (65))*

```sql
CREATE PROCEDURE FindTopHeadshotsCount(
    @n INT
)
AS
BEGIN
    SET NOCOUNT ON;

    WITH RankedHeadshots AS (
        SELECT
            L.Dev_ID,
            L.difficulty,
            L.headshots_count,
            ROW_NUMBER() OVER (PARTITION BY L.Dev_ID ORDER BY L.headshots_count ASC) AS RN
        FROM
            Level_Details L
    )
    SELECT TOP (@n)
        Dev_ID,
        difficulty,
        headshots_count,
        RN
    FROM
        RankedHeadshots
    WHERE
        RN<= @n;
END;
```

100 %

Messages

```
Commands completed successfully.
```

100 %

Query executed successfully. (local) (16.0 RTM) DESKTOP-LBV1D6M\dell (65) omar 00:00:00 0 rows

SQLQuery3.sql - (l...-LBV1D6M\dell (65))*

```sql
EXEC FindTopHeadshotsCount @n = 5; -- Example: Retrieve top 5 headshots_count for each Dev_ID
```

100 %

Results | Messages

	Dev_ID	difficulty	headshots_count	RN
1	bd_013	Medium	4	1
2	bd_013	Medium	8	2
3	bd_013	Medium	10	3
4	bd_013	Difficult	11	4
5	bd_013	Low	11	5

THANK YOU