Gaming Analysis with SQL

Internship Project by Harsh Garg with Mentorness Batch –MIP-DA-06

Content

- About the Project
- Dataset Description
- Import the Data
- Prepare and Modify the Data
- Analysis SQL Problem Statement
- What I learn

About the Project

- In this internship, I was working with a dataset related to a game. The dataset includes two tables:

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

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 must 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.

Import the Data

- Create Database

```
CREATE DATABASE Gaming_Analysis;
```

90 %

Messages

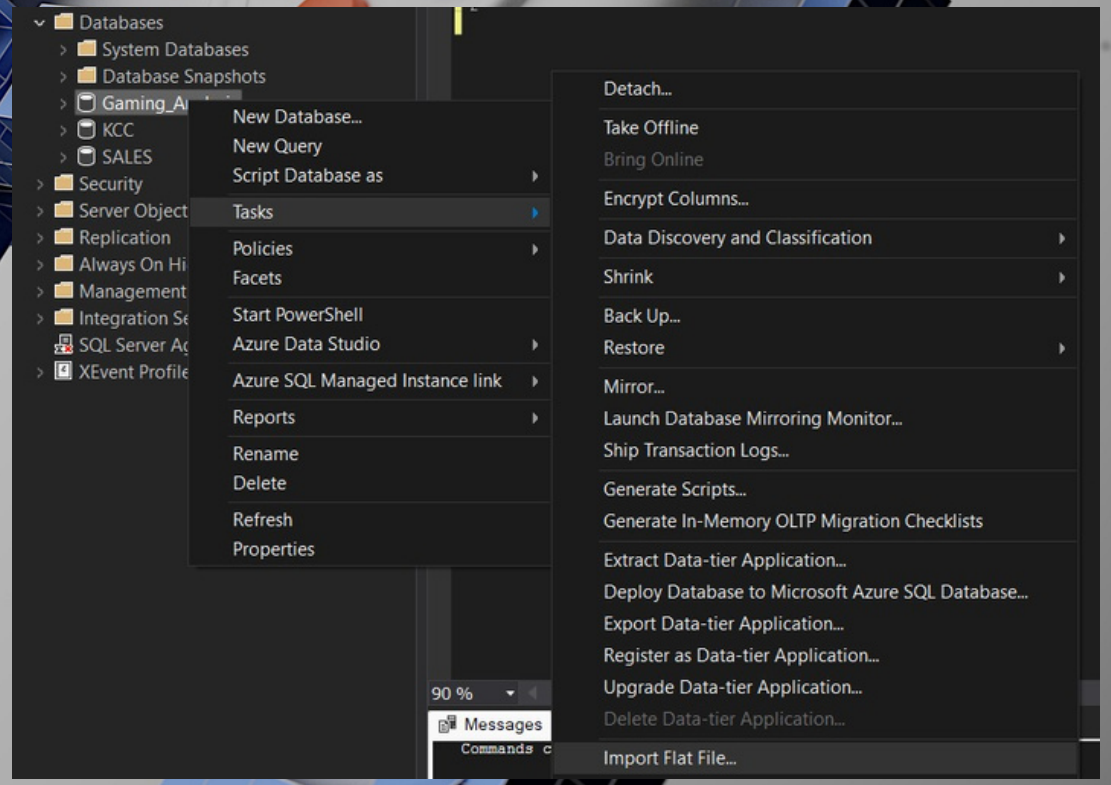
```
Commands completed successfully.

Completion time: 2024-04-16T00:41:08.6950693+02:00
```

90 %

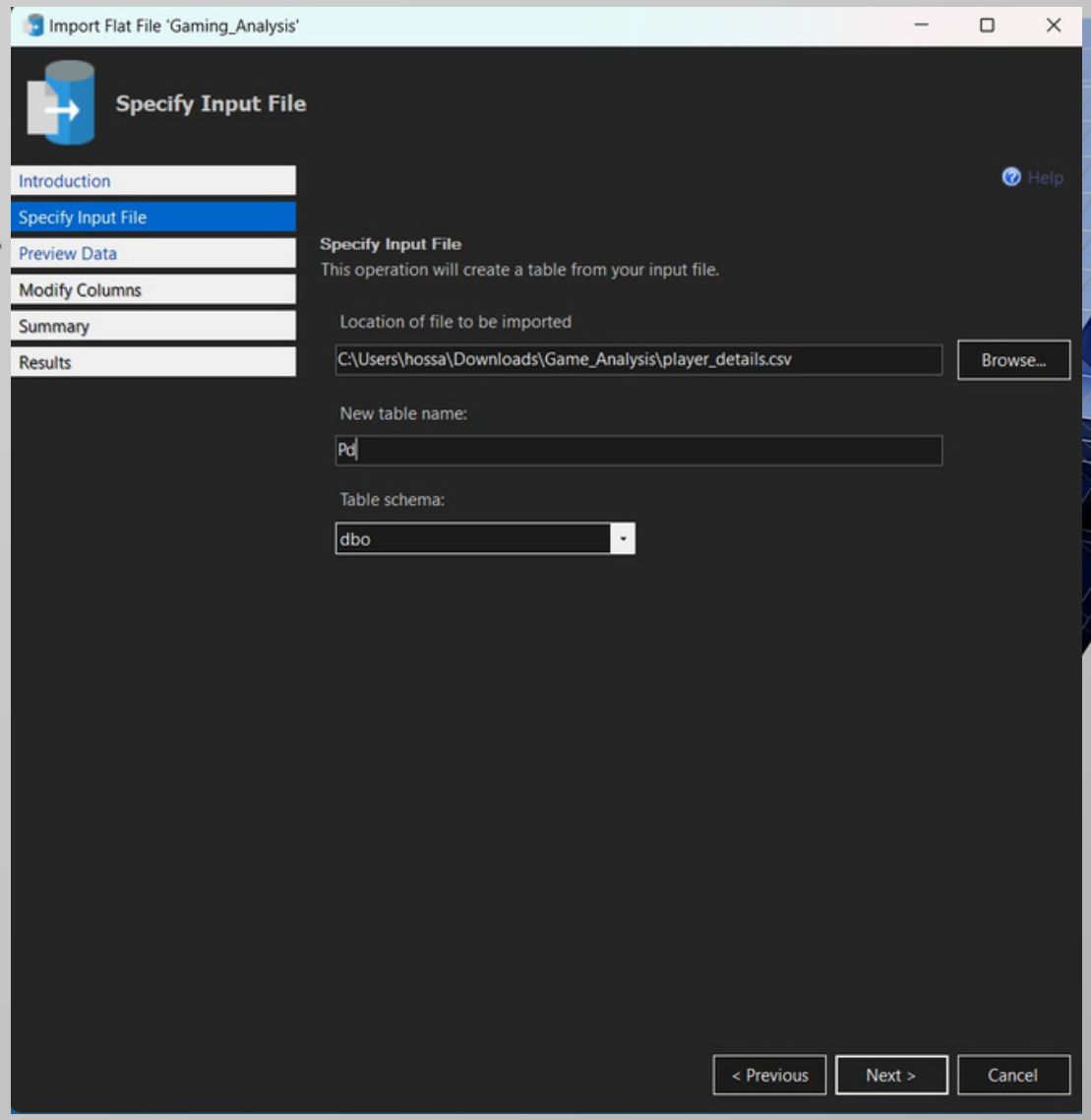
Query executed successfully.

- Import as flat file



Import the Data

Browse the File



Prepare and Modify the Data

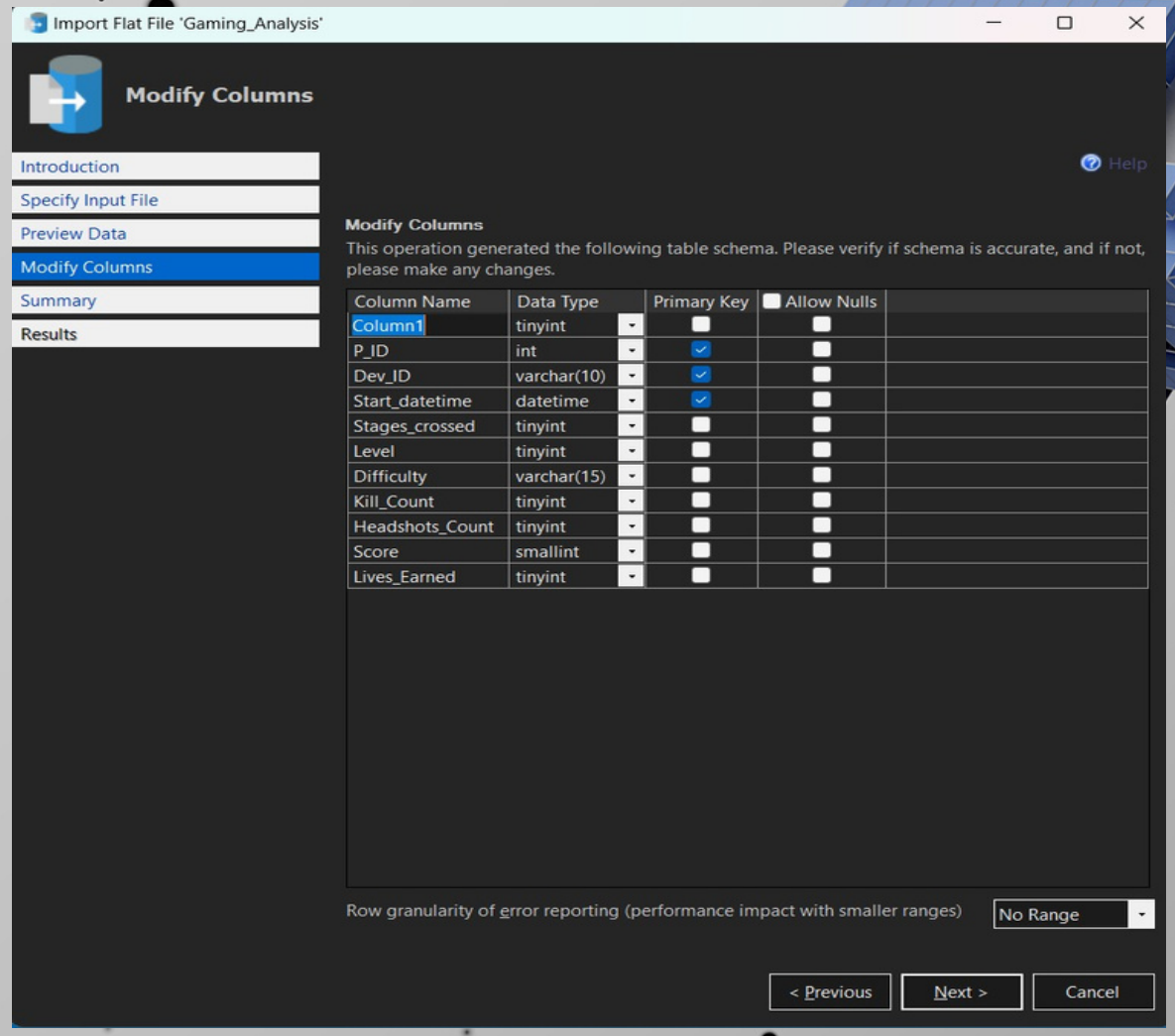
	Column Name	Data Type	Allow Nulls
▶	Column1	tinyint	☐
	P_ID	smallint	☐
	PName	nvarchar(50)	☐
	L1_Status	bit	☐
	L2_Status	bit	☐
	L1_Code	nvarchar(50)	☑
	L2_Code	nvarchar(50)	☑
			☐

TASKS:
1.Delete Column1
2.Change Datatype for L1_Status and L1_Status to varchar(30)
3.Change Datatype for P_ID to int primary key

	Column Name	Data Type	Allow Nulls
▶	Column1	tinyint	☐
	P_ID	smallint	☐
	Dev_ID	nvarchar(50)	☐
	TimeStamp	datetime2(7)	☐
	Stages_crossed	tinyint	☐
	[Level]	tinyint	☐
	Difficulty	nvarchar(50)	☐
	Kill_Count	tinyint	☐
	Headshots_Count	tinyint	☐
	Score	smallint	☐
	Lives_Earned	tinyint	☐
			☐

Tasks:
1.Delete Column1
2.Change Column name for TimeStamp to Start_datetime and datatype to datetime
3.Change Datatype for P_ID, Dev_ID, and Start_datatime to primary key
4.Change Datatype for Dev_ID to varchar(10)
5.Change Datatype for Difficulty to varchar(15)

Modify the Data in Drop down menu



Column Name	Data Type	Allow Nulls
P_ID	int	
Dev_ID	varchar(10)	
Start_datetime	datetime	
Stages_crossed	tinyint	
[Level]	tinyint	
Difficulty	varchar(15)	
Kill_Count	tinyint	
Headshots_Count	tinyint	
Score	smallint	
Lives_Earned	tinyint	

Modify the Data in SQL Queries

```
USE Gaming_Analysis
ALTER TABLE Pd DROP COLUMN Column1;
ALTER TABLE Pd ADD CONSTRAINT P_ID PRIMARY KEY(P_ID);
ALTER TABLE Pd ALTER COLUMN L1_Status VARCHAR(30);
ALTER TABLE Pd ALTER COLUMN L2_Status VARCHAR(30);
ALTER TABLE Ld DROP COLUMN Column1;
```

90 %

Messages

```
Commands completed successfully.

Completion time: 2024-04-16T21:56:41.2880061+02:00
```

90 %

Query executed successfully.

Column Name	Data Type	Allow Nulls
P_ID	int	☐
PName	nvarchar(50)	☐
L1_Status	varchar(30)	☑
L2_Status	varchar(30)	☑
L1_Code	nvarchar(50)	☑
L2_Code	nvarchar(50)	☑
		☐

Analysis SQL Problem Statement

```sql
SELECT
      Ld.P_ID,
      Ld.Dev_ID,
      Pd.PName,
      Ld.Difficulty
FROM
      Ld INNER JOIN Pd ON (Ld.P_ID = Pd.P_ID)
WHERE
      Level = 0;
```

99 %

Results | Messages

	P_ID	Dev_ID	PName	Difficulty
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.

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

Analysis SQL Problem Statement

```sql
USE Gaming_Analysis
SELECT
      Pd.L1_Code,
      AVG(Kill_Count) AS avg_kill_count
FROM
      Ld INNER JOIN Pd ON (Ld.P_ID = Pd.P_ID)
WHERE
      Lives_Earned = 2 AND
      Stages_crossed >= 3
GROUP BY
      L1_Code;
```

99 %

Results | Messages

	L1_Code	avg_kill_count
1	bulls_eye	22
2	speed_blitz	19
3	war_zone	19

Query executed successfully.

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

Analysis SQL Problem Statement

```sql
USE Gaming_Analysis
SELECT
      SUM(Stages_crossed) AS total_stages_crossed,
      Difficulty
FROM
     Ld
WHERE
     Level = 2 AND
     Dev_ID like 'zm%'
GROUP BY
     Difficulty
ORDER BY
     total_stages_crossed DESC;
```

99 %

Results | Messages

	total_stages_crossed	Difficulty
1	46	Difficult
2	35	Medium
3	15	Low

Query executed successfully.

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.

Analysis SQL Problem Statement

```sql
USE Gaming_Analysis
SELECT
      P_ID,
      COUNT(DISTINCT Start_datetime) AS unique_dates
FROM
     Ld
GROUP BY
     P_ID
HAVING
     COUNT(DISTINCT Start_datetime) > 1;
```

99 %

Results | Messages

	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

Query executed successfully.

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

Analysis SQL Problem Statement

```sql
USE Gaming_Analysis
SELECT
	P_ID,
	Level,
	SUM(Kill_Count) AS total_kill_count
FROM
	Ld
WHERE
	Kill_Count > (SELECT AVG(Kill_Count) FROM Ld WHERE Difficulty = 'Medium')
GROUP BY
	P_ID,
	Level;
```

99 %

Results | Messages

	P_ID	Level	total_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
15	590	1	24
16	632	1	28
17	656	1	37

Query executed successfully.

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

Analysis SQL Problem Statement

```sql
USE Gaming_Analysis
SELECT
      Level,
      COALESCE (L1_Code, L2_Code) AS Level_code,
      SUM(Lives_Earned) AS total_lives_earned
FROM
      Ld INNER JOIN Pd ON (Ld.P_ID = Pd.P_ID)

WHERE
      Level <> 0
GROUP BY
      Level,
      L1_Code,
      L2_Code
ORDER BY
      Level ASC;
```

99 %

Results | Messages

	Level	Level_code	total_lives_earned
1	1	bulls_eye	3
2	1	bulls_eye	1
3	1	bulls_eye	1
4	1	leap_of_faith	0
5	1	speed_blitz	0
6	1	speed_blitz	4
7	1	speed_blitz	3
8	1	war_zone	4
9	1	war_zone	0
10	1	war_zone	7
11	2	bulls_eye	6
12	2	bulls_eye	8
13	2	speed_blitz	6
14	2	speed_blitz	14

Query executed successfully.

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

Analysis SQL Problem Statement

```sql
USE Gaming_Analysis
SELECT
      TOP 3 (Score),
      Dev_ID,
      ROW_NUMBER() OVER(PARTITION BY Dev_ID ORDER BY Score) AS rownumber,
      Difficulty
FROM
      Ld
GROUP BY
      Dev_ID,
      Score,
      Difficulty;
```

99 %

Results | Messages

	Score	Dev_ID	rownumber	Difficulty
1	100	bd_013	1	Difficult
2	540	bd_013	2	Low
3	590	bd_013	3	Medium

Query executed successfully.

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.

Analysis SQL Problem Statement

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

```
USE Gaming_Analysis
SELECT
      MIN(Start_datetime) AS first_login_datetime,
      Dev_ID
FROM
      Ld
GROUP BY
      Dev_ID;
```

99 %

Results | Messages

	first_login_datetime	Dev_ID
1	2022-10-11 02:23:00.000	bd_013
2	2022-10-11 18:45:00.000	bd_015
3	2022-10-12 07:30:00.000	bd_017
4	2022-10-11 05:20:00.000	rf_013
5	2022-10-11 19:34:00.000	rf_015
6	2022-10-11 09:28:00.000	rf_017
7	2022-10-12 23:19:00.000	wd_019
8	2022-10-11 13:00:00.000	zm_013
9	2022-10-11 14:05:00.000	zm_015
10	2022-10-11 14:33:00.000	zm_017

Query executed successfully.

Analysis SQL Problem Statement

```
USE Gaming_Analysis
SELECT
      TOP 5 (Score),
      Difficulty,
      Dev_ID,
      RANK() OVER(PARTITION BY Dev_id ORDER BY Score) AS 'Rank'
FROM
      Ld;
```

99 %

Results | Messages

	Score	Difficulty	Dev_ID	Rank
1	100	Difficult	bd_013	1
2	100	Difficult	bd_013	1
3	540	Low	bd_013	3
4	590	Medium	bd_013	4
5	1100	Medium	bd_013	5

Query executed successfully.

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

Analysis SQL Problem Statement

```
USE Gaming_Analysis
SELECT
      P_ID,
      Dev_ID,
      MIN(Start_datetime) AS first_login_datetime
FROM
      Ld
GROUP BY
      P_ID,
      Dev_ID;
```

99 %

Results | Messages

	P_ID	Dev_ID	first_login_datetime
1	211	bd_013	2022-10-12 18:30:00.000
2	211	bd_017	2022-10-12 13:23:00.000
3	211	rf_013	2022-10-13 05:36:00.000
4	211	rf_017	2022-10-15 11:41:00.000
5	211	zm_015	2022-10-13 22:30:00.000
6	211	zm_017	2022-10-14 08:56:00.000
7	224	bd_013	2022-10-15 05:30:00.000
8	224	bd_015	2022-10-14 08:21:00.000
9	224	rf_017	2022-10-14 01:15:00.000
10	242	bd_013	2022-10-13 01:14:00.000
11	242	zm_015	2022-10-14 04:38:00.000
12	292	rf_013	2022-10-12 04:29:00.000
13	292	rf_015	2022-10-15 10:19:00.000
14	296	zm_015	2022-10-14 19:35:00.000
15	296	zm_017	2022-10-14 15:15:00.000
16	300	bd_013	2022-10-11 19:19:00.000
17	300	rf_013	2022-10-11 05:20:00.000

Query executed successfully.

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.

Analysis SQL Problem Statement

```sql
-- a) window function--
USE Gaming_Analysis
SELECT
	P_ID,
	CAST(Start_datetime AS date) AS 'Date',
	SUM(Kill_Count) OVER(PARTITION BY P_ID ORDER BY Start_datetime) AS total_kill_count
FROM
	Ld;

-- b) without window function.
USE Gaming_Analysis
SELECT
	P_ID,
	CAST(Start_datetime AS date) AS 'Date',
	(SELECT SUM(Kill_Count) FROM Ld ld2 WHERE ld2.P_ID = Ld.P_ID AND ld2.Start_datetime <= Ld.Start_datetime) AS total_kill_count
FROM
	Ld;
```

99 %

Results | Messages

	P_ID	Date	total_kill_count
1	211	2022-10-12	20
2	211	2022-10-12	45
3	211	2022-10-13	75
4	211	2022-10-13	89
5	211	2022-10-14	98
6	211	2022-10-15	113

	P_ID	Date	total_kill_count
1	211	2022-10-12	45
2	211	2022-10-12	20
3	211	2022-10-13	75
4	211	2022-10-15	113
5	211	2022-10-13	89
6	211	2022-10-14	98

Query executed successfully. HOSSAM (16.0 RTM) HOSSAM\hos

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

Analysis SQL Problem Statement

```
USE Gaming_Analysis
SELECT
        P_ID,
        Start_Datetime,
        SUM(Stages_crossed) OVER(PARTITION BY P_ID ORDER BY Start_Datetime ASC) AS cumulative_sum_of_stages_crossed
FROM
        Ld;
```

99 %

Results | Messages

	P_ID	Start_Datetime	cumulative_sum_of_stages_crossed
1	211	2022-10-12 13:23:00.000	4
2	211	2022-10-12 18:30:00.000	9
3	211	2022-10-13 05:36:00.000	14
4	211	2022-10-13 22:30:00.000	19
5	211	2022-10-14 08:56:00.000	26
6	211	2022-10-15 11:41:00.000	34
7	224	2022-10-14 01:15:00.000	7
8	224	2022-10-14 08:21:00.000	12
9	224	2022-10-15 05:30:00.000	22
10	224	2022-10-15 13:43:00.000	26
11	242	2022-10-13 01:14:00.000	6
12	242	2022-10-14 04:38:00.000	14
13	292	2022-10-12 04:29:00.000	4

Query executed successfully. HOSSAM (1

12) Find the cumulative sum of stages crossed over `start_datetime` for each `P_ID`, excluding t

Analysis SQL Problem Statement

```sql
USE Gaming_Analysis
SELECT
        TOP 3 SUM(Score) AS highest_scores,
        P_ID,
        Dev_ID
FROM
        Ld
GROUP BY
        P_ID,
        Dev_ID
ORDER BY
        SUM(Score) DESC;
```

99 %

Results | Messages

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

Query executed successfully.

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

Analysis SQL Problem Statement

```sql
USE Gaming_Analysis
SELECT
	P_ID,
	SUM(Score) AS total_scores
FROM
	Ld
GROUP BY
	P_ID
HAVING
	SUM(Score) > 0.5*(SELECT AVG(Score) FROM Ld);
```

99 %

Results | Messages

	P_ID	total_scores
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

Query executed successfully.

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

Analysis SQL Problem Statement

```sql
CREATE PROCEDURE top_n_headshots_count
                @n INT
AS
BEGIN
SELECT
      Dev_ID,
      headshots_count,
      difficulty,
      ROW_NUMBER() OVER (PARTITION BY Dev_ID ORDER BY headshots_count) AS Row_Number
FROM
     Ld
ORDER BY
  Dev_ID,
  Row_Number
OFFSET
  0 ROWS
FETCH NEXT
  @n ROWS ONLY;
END

EXEC    [dbo].[top_n_headshots_count] @n = 4;
```

99 %

Results | Messages

	Dev_ID	headshots_count	difficulty	Row_Number
1	bd_013	4	Medium	1
2	bd_013	8	Medium	2
3	bd_013	10	Medium	3
4	bd_013	11	Difficult	4

Query executed successfully.

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.

Analysis SQL Problem Statement

```sql
CREATE FUNCTION Total_Score
(
    @player_id AS INT
)
RETURNS INT
AS
BEGIN
    DECLARE @total_score AS INT;

    SELECT @total_score = SUM(Score)
    FROM Ld
    WHERE P_ID = @player_id;

    RETURN @total_score;
END;

SELECT dbo.Total_Score (644) AS Total_Score;
```

99 %

Results | Messages

	Total_Score
1	2250

Query executed successfully.

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

What I learn

- SQL Joins
- SQL GROUP BY and HAVING
- Clause A Nested Query
- COALESCE and CAST Function
- Row Number and Rank Function
- Minimum and Top Function
- SQL Declare Variable OFFSET-
- FETCH clause

Thank You