PRESENTED BY AHLAM SHAQFA

MENTORNESS PROJECT1

DECODE GAMING BEHAVIOUR

MENTOR NESS

TABLE OF CONTENT

- Objectives of decoding.
- Queries used.
- Conclusion.

OBJECTIVES:

- Two distinct datasets containing player gaming histories, including IDs, kill_totals, etc., are provided to us.
- We must respond to fifteen inquiries that provide us with information about their records and behavior.
- We will be utilizing MySQL Workbench for Data Analysis. After importing datasets into a new database, the procedure starts.

Sample of the Two Datasets

```
300 •   SELECT * FROM player_details;
301
```

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

	column1	p_id	pname	l1_status	l2_status	l1_code	l2_code
▶	0	656	sloppy-denim-wolfhound	1	0	war_zone	
	1	358	skinny-grey-quetzal	0	0		
	2	296	silly-taupe-ray	1	0	war_zone	
	3	644	randy-turquoise-scorpion	1	1	speed_blitz	cosmic_vision
	4	320	chewy-harlequin-gharial	0	0		
	5	632	dorky-heliotrope-barracuda	1	1	speed_blitz	splippery_slope
	6	428	leaky-magnolia-iguana	1	0	leap_of_faith	
	7	429	flabby-firebrick-bee	1	1	speed_blitz	cosmic_vision
	8	310	gloppy-tomato-wasp	1	1	war_zone	splippery_slope
	9	211	breezy-indigo-starfish	1	1	war_zone	splippery_slope
	10	319	chummy-flax-crab	1	0	speed_blitz	

```
302 •   SELECT * FROM level_details2;
303
304
305
306
```

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

	column1	p_id	dev_id	TimeStamp	stages_crossed	Level	difficulty	kill_count	headshots_count	score	lives_earned
▶	0	644	zm_015	2022-10-11 14:05:08	3	1	Medium	11	5	350	1
	1	644	rf_015	2022-10-11 19:34:25	1	1	Low	7	2	150	0
	2	644	bd_017	2022-10-12 23:52:18	6	2	Medium	24	16	1750	2
	3	656	rf_013	2022-10-15 18:12:50	7	0	Medium	15	8	880	0
	4	656	bd_015	2022-10-13 22:19:45	4	1	Low	19	13	1450	0
	5	656	rf_017	2022-10-14 07:32:00	2	1	Difficult	3	1	280	1
	6	656	bd_013	2022-10-11 17:47:09	10	1	Low	18	16	2210	3
	7	296	zm_017	2022-10-14 15:15:15	2	1	Difficult	7	3	1040	0
	8	296	zm_015	2022-10-14 19:35:49	4	1	Medium	4	0	100	0
	9	632	bd_013	2022-10-12 16:30:30	5	0	Difficult	45	30	100	0
	10	632	rf_013	2022-10-12 19:36:40	5	1	Medium	28	25	100	1

This is the sample of the content of the data sets, for the first columns in both are named column1 it indicates that they are considered to be as index , and we could change their names, I chose not to do because we will not use them so just ignored.

MENTOR NESS

Query 1 :

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

```sql
use game;

/*Q1: Extract `P_ID`, `Dev_ID`, `PName`, and `Difficulty_level` of all players at Level 0*/

SELECT
    player_details.p_id,
    player_details.pname,
    level_details2.dev_id,
    level_details2.difficulty
FROM
    Player_Details
        JOIN
    level_details2 ON Player_Details.p_id = level_details2.p_id
WHERE
    level_details2.level = 0;
```

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

p_id	pname	dev_id	difficulty
656	sloppy-denim-wolfhound	rf_013	Medium
358	skinny-grey-quetzal	zm_013	Medium
358	skinny-grey-quetzal	zm_017	Low
632	dorky-heliotrope-barracuda	bd_013	Difficult
429	flabby-firebrick-bee	bd_013	Medium
310	gloppy-tomato-wasp	bd_015	Difficult

MENTOR NESS

Query 2 :

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

```
17
18  /*Q2: Find `Level1_code`wise average `Kill_Count` where `lives_earned` is 2, and at least 3
19  stages are crossed*/
20
21  SELECT
22      player_details.L1_code,
23      AVG(level_details2.kill_count) AS avg_kill_count
24  FROM
25      player_details
26          JOIN
27      level_details2 ON Player_details.p_id = level_details2.p_id
28  WHERE
29      lives_earned = 2 AND stages_crossed >= 3
30  GROUP BY player_details.l1_code
31
32
```

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

L1_code	avg_kill_count
speed_blitz	19.3333
war_zone	19.2857
bulls_eye	22.2500

MENTOR NESS

Query 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.

```
32
33  /*Q3:  Find the total number of stages crossed at each difficulty level for Level 2 with players
34  using `zm_series` devices. Arrange the result in decreasing order of the total number of
35  stages crossed.*/
36
37  SELECT
38      level_details2.difficulty,
39      SUM(level_details2.stages_crossed) AS total_stages_crossed
40  FROM
41      level_details2
42  WHERE
43      level_details2.`Level` = 2
44          AND level_details2.dev_id IN ('zm_013' , 'zm_015', 'zm_017')
45  GROUP BY level_details2.difficulty
46  ORDER BY total_stages_crossed DESC;
47
```

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

difficulty	total_stages_crossed
Difficult	46
Medium	35
Low	15

MENTOR NESS

Query 4 :

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

```
53
54  /*Q4:  Extract `P_ID` and the total number of unique dates for those players who have played
55  games on multiple days.*/
56
57  SELECT
58      level_details2.p_id,
59      COUNT(DISTINCT DATE(level_details2.`TimeStamp`)) AS total_unique_dates
60  FROM
61      level_details2
62  GROUP BY
63      level_details2.p_id
64  HAVING
65      total_unique_dates > 1;
66
67
68
```

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

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

MENTOR NESS

Query 5 :

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

```sql
/* Q5: Find `P_ID` and levelwise sum of `kill_counts` where `kill_count` is greater than the
average kill count for Medium difficulty.*/
SELECT
    ld2.P_ID, ld2.level, SUM(ld2.kill_count) AS total_kill_count
FROM
    level_details2 ld2
        JOIN
    (SELECT
        level, AVG(kill_count) AS avg_kill_count
    FROM
        level_details2
    WHERE
        difficulty = 'Medium'
    GROUP BY level) AS avg_table ON ld2.level = avg_table.level
        AND ld2.difficulty = 'Medium'
WHERE
    ld2.kill_count > avg_table.avg_kill_count
GROUP BY ld2.P_ID , ld2.level;

```

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

P_ID	level	total_kill_count
644	2	24
656	0	15
632	1	28
632	2	23
429	0	14

MENTOR NESS

Query 6 :

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

```
83      /*Q:6 Find `Level` and its corresponding `Level_code`wise sum of lives earned, excluding Level 0.
84       * Arrange in ascending order of level.*/
85  •  SELECT
86          ld.`Level`,
87          CONCAT(pd.L1_Code, '/', pd.L2_Code) AS Level_code,
88          SUM(ld.Lives_Earned) AS total_lives_earned
89      FROM
90          Player_Details pd
91              JOIN
92          level_details2 ld ON pd.P_ID = ld.P_ID
93      WHERE
94          ld.`Level` > 0
95      GROUP BY ld.`Level` , CONCAT(pd.L1_Code, '/', pd.L2_Code)
96      ORDER BY ld.`Level` ASC;
97
```

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

Level	Level_code	total_lives_earned
1	bulls_eye/	3
1	bulls_eye/cosmic_vision	1
1	bulls_eye/resurgence	1
1	leap_of_faith/	0
1	speed_blitz/	0

MENTOR NESS

Query 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.

```
97
98      /*Q7: Find the top 3 scores based on each `Dev_ID` and rank them in increasing order using
99  `Row_Number`. Display the difficulty as well.*/
100 •   SELECT
101         Dev_ID,
102         score,
103         difficulty,
104         row_num
105     FROM (
106         SELECT
107             Dev_ID,
108             score,
109             difficulty,
110             ROW_NUMBER() OVER (PARTITION BY Dev_ID ORDER BY score DESC) AS row_num
111         FROM
112             level_details2
113     ) AS ranked_scores
114     WHERE
115         row_num <= 3
116     ORDER BY Dev_ID, row_num;
```

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

Dev_ID	score	difficulty	row_num
bd_013	5300	Difficult	1
bd_013	4570	Difficult	2
bd_013	3370	Difficult	3
bd_015	5300	Difficult	1
bd_015	3200	Low	2

MENTOR NESS

Query 8 :

Find the `first_login` datetime for each device ID.

```
118
119  /*Q8: Find the `first_login` datetime for each device ID.
120   */
121  SELECT
122      Dev_ID, MIN(`TimeStamp`) AS first_login
123  FROM
124      level_details2
125  GROUP BY Dev_ID;
126
```

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

Dev_ID	first_login
zm_015	2022-10-11 14:05:08
rf_015	2022-10-11 19:34:25
bd_017	2022-10-12 07:30:18
rf_013	2022-10-11 05:20:40
bd_015	2022-10-11 18:45:55

MENTOR NESS

Query 9 :

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

```
127
128  /*Q9:Find the top 5 scores based on each difficulty level and rank them in increasing order
129  using `Rank`. Display `Dev_ID` as well*/
130  SELECT
131      Dev_ID,
132      score,
133      difficulty,
134      Rank_score
135  FROM (
136      SELECT
137          Dev_ID,
138          score,
139          difficulty,
140          RANK() OVER (PARTITION BY difficulty ORDER BY score DESC) AS Rank_score
141      FROM
142          level_details2
143  ) AS ranked_scores
144  WHERE
145      Rank_score <= 5
146  ORDER BY difficulty, Rank_score;
```

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

Dev_ID	score	difficulty	Rank_score
zm_017	5500	Difficult	1
zm_017	5500	Difficult	1
bd_015	5300	Difficult	3
bd_013	5300	Difficult	3
rf_017	5140	Difficult	5

MENTOR NESS

Query 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.

```
148
149  /*Q10: Find the device ID that is first logged in (based on `start_datetime`) for each player
150  (`P_ID`). Output should contain player ID, device ID, and first login datetime.*/
151  SELECT
152      P_ID,
153      Dev_ID,
154      `TimeStamp` AS first_login_datetime
155  FROM (
156      SELECT
157          P_ID,
158          Dev_ID,
159          `TimeStamp`,
160          ROW_NUMBER() OVER (PARTITION BY P_ID ORDER BY `TimeStamp` ASC) AS rn
161      FROM
162          level_details2
163  ) AS ranked_logins
164  WHERE
165      rn = 1;
```

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

P_ID	Dev_ID	first_login_datetime
211	bd_017	2022-10-12 13:23:45
224	rf_017	2022-10-14 01:15:56
242	bd_013	2022-10-13 01:14:29
292	rf_013	2022-10-12 04:29:45
296	zm_017	2022-10-14 15:15:15

MENTOR NESS

Query 11 – a :

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

```sql
/*Q11-a: For each player and date, determine how many `kill_counts` were played by the player
so far.
a) Using window functions*/

SELECT
    P_ID,
    TimeStamp,
    total_kill_counts_so_far
FROM (
    SELECT
        P_ID,
        `TimeStamp`,
        SUM(kill_count) OVER (PARTITION BY P_ID ORDER BY `TimeStamp`) AS total_kill_counts_so_far,
        ROW_NUMBER() OVER (PARTITION BY P_ID, DATE(`TimeStamp`) ORDER BY `TimeStamp`) AS rn
    FROM
        level_details2
) AS ranked_kills
WHERE
    rn = 1;
```

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

P_ID	TimeStamp	total_kill_counts_so_far
211	2022-10-12 13:23:45	20
211	2022-10-13 05:36:15	75
211	2022-10-14 08:56:24	98
211	2022-10-15 11:41:19	113
224	2022-10-14 01:15:56	20

MENTOR NESS

Query 11 – b :

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

```
186  /*Q11-b: For each player and date, determine how many `kill_counts` were played by the player
187  so far.
188  b) Without window functions*/
189   SELECT
190      ld.P_ID,
191      (ld.`TimeStamp`) AS date,
192      (SELECT
193              SUM(ld_inner.kill_count)
194          FROM
195              level_details2 ld_inner
196          WHERE
197              ld_inner.P_ID = ld.P_ID
198                  AND DATE(ld_inner.`TimeStamp`) <= DATE(ld.`TimeStamp`)) AS total_kill_counts_so_far
199  FROM
200      level_details2 ld;
```

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

P_ID	date	total_kill_counts_so_far
644	2022-10-11 14:05:08	18
644	2022-10-11 19:34:25	18
644	2022-10-12 23:52:18	42
656	2022-10-15 18:12:50	55
656	2022-10-13 22:19:45	37

MENTOR NESS

Query 12 :

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

```
201
202  /*Q12: Find the cumulative sum of stages crossed over `start_datetime` for each `P_ID`,
203  excluding the most recent `start_datetime`*/
204
205  SELECT
206      ld.P_ID,
207      ld.`TimeStamp`,
208      SUM(ld.stages_crossed) AS cumulative_stages_crossed
209  FROM
210      level_details2 ld
211          JOIN
212      (SELECT
213          P_ID, MAX(level_details2.`TimeStamp`) AS max_start_datetime
214      FROM
215          level_details2
216      GROUP BY P_ID) max_start ON ld.P_ID = max_start.P_ID
217          AND ld.`TimeStamp` < max_start.max_start_datetime
218  GROUP BY ld.P_ID , ld.`TimeStamp`
219  ORDER BY ld.P_ID , ld.`TimeStamp`;
220
```

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

P_ID	TimeStamp	cumulative_stages_crossed
211	2022-10-12 13:23:45	4
211	2022-10-12 18:30:30	5
211	2022-10-13 05:36:15	5
211	2022-10-13 22:30:18	5
211	2022-10-14 08:56:24	7

MENTOR NESS

Query 13 :

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

```sql
/*Q13: Extract the top 3 highest sums of scores for each `Dev_ID` and the corresponding `P_ID`*/
WITH RankedScores AS (
  SELECT
        P_ID,
        Dev_ID,
        SUM(score) AS total_score,
        ROW_NUMBER() OVER (PARTITION BY Dev_ID ORDER BY SUM(score) DESC) AS row_num
  FROM
       level_details2
  GROUP BY
        P_ID, Dev_ID
)
SELECT
    P_ID,
    Dev_ID,
    total_score
FROM
    RankedScores
WHERE
    row_num <= 3;
```

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

P_ID	Dev_ID	total_score
224	bd_013	9870
310	bd_013	3370
211	bd_013	3200
310	bd_015	5300
683	bd_015	3200

MENTOR NESS

Query 14 :

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

```sql
/*Q14:Find players who scored more than 50% of the average score, scored by the sum of
scores for each `P_ID`*/
 WITH PlayerAvgScore AS (
 SELECT
        P_ID,
        AVG(score) AS avg_score
 FROM
        level_details2
 GROUP BY
        P_ID
)
 SELECT
     ld.P_ID,
     ld.score,
     pas.avg_score
 FROM
     level_details2 ld
 JOIN
     PlayerAvgScore pas ON ld.P_ID = pas.P_ID
 WHERE
     ld.score > 0.5 * pas.avg_score;
```

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

P_ID	score	avg_score
644	1750	750.0000
656	880	1205.0000
656	1450	1205.0000
656	2210	1205.0000
296	1040	570.0000

MENTOR NESS

Query 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

```
68 •  drop procedure if exists TopHeadshotsPerDevice;
69    DELIMITER $$
70 •  CREATE PROCEDURE TopHeadshotsPerDevice (IN n INT)
71    BEGIN
72        SELECT *
73        FROM (
74            SELECT t.P_ID, t.Dev_ID, t.headshots_count, t.difficulty,
75                   ROW_NUMBER() OVER(PARTITION BY t.Dev_ID ORDER BY t.headshots_count ASC) AS rn
76            FROM (
77                SELECT ld.P_ID, ld.Dev_ID, ld.headshots_count, ld.difficulty
78                FROM level_details2 ld
79                JOIN (
80                    SELECT Dev_ID, MAX(headshots_count) AS max_headshots
81                    FROM level_details2
82                    GROUP BY Dev_ID
83                ) max_h ON ld.Dev_ID = max_h.Dev_ID AND ld.headshots_count = max_h.max_headshots
84            ) t
85        ) ranked
86        WHERE rn <= n;
87    END$$
88    DELIMITER ;
```

MENTOR NESS

Query 15 :

Calling the procedure

```
290
291 •   CALL TopHeadshotsPerDevice(5);
292
293
294
295
296
297
```

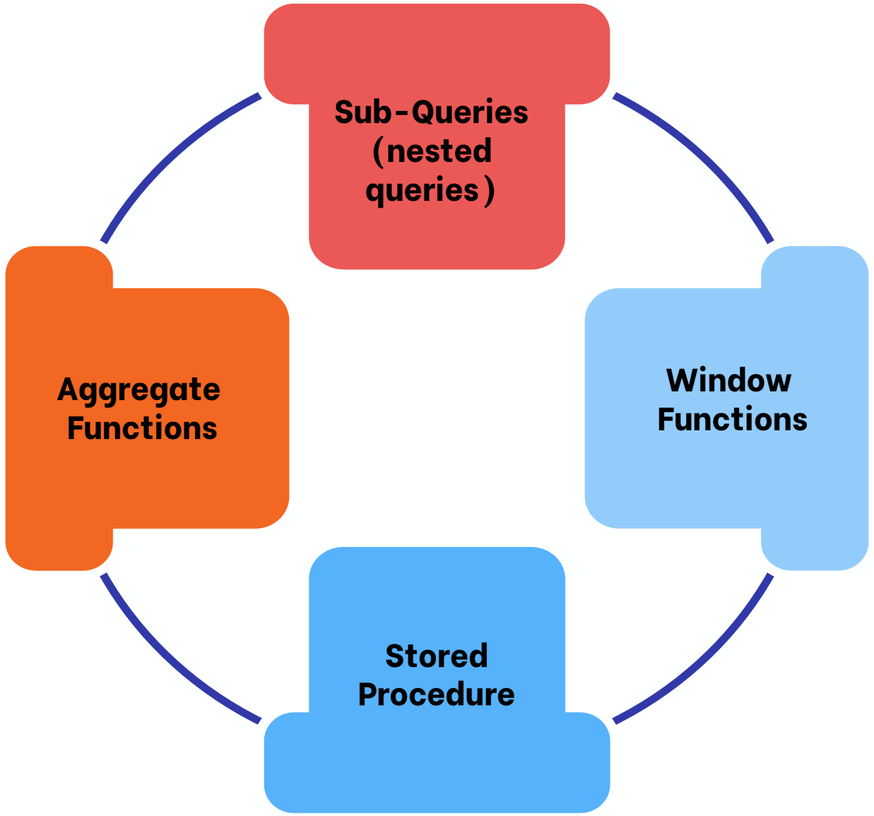
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

P_ID	Dev_ID	headshots_count	difficulty	rn
632	bd_013	30	Difficult	1
663	bd_013	30	Difficult	2
310	bd_015	30	Difficult	1
224	bd_015	30	Difficult	2
590	bd_017	18	Low	1

MENTOR NESS

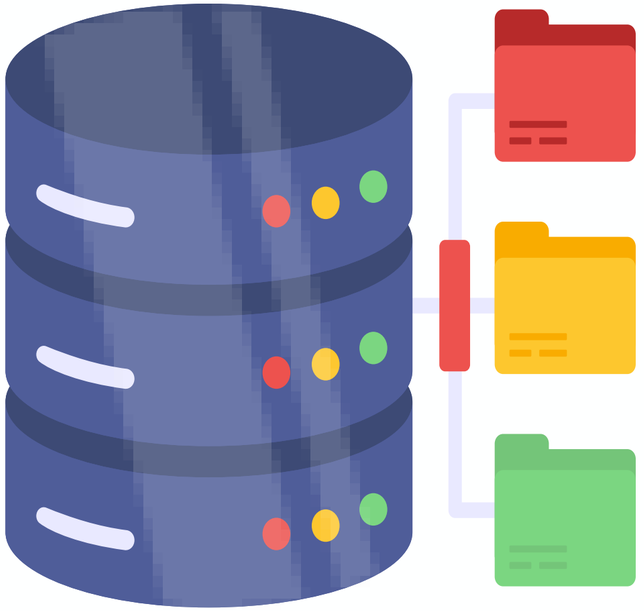
Conclusion

We note that the main SQL concepts this analysis focuses on are :



MENTOR NESS

THANK YOU!



MENTOR NESS