GAME ANLYSIS

Let's start!

Presented by Kanchan Prajapati

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_= 0;
```

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

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

Q2) FIND LEVEL1_CODE WISE AVG_KILL_COUNT WHERE LIVES_EARNED IS 2 AND ATLEAST 3 STAGES ARE CROSSED

```
select player_details.L1_code ,
avg(level_Details2.Kill_count) as Avg_kill_count
from player_details
join level_Details2
on player_details.P_ID=level_Details2.P_ID
where
Lives_Earned=2
and
Stages_crossed >=3
group by player_details.L1_Code;
```

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

L1_code	Avg_kill_count
war_zone	19.2857
bulls_eye	22.2500
speed_blitz	19.3333

Q3) FIND THE TOTAL NUMBER OF STAGES CROSSED AT EACH DIFFUCULTY LEVEL WHERE FOR LEVEL2 WITH PLAYERS USE ZM_SERIES DEVICES. ARRANGE THE RESULT IN DECSREASING ORDER OF TOTAL NUMBER OF STAGES CROSSED

```sql
select Difficulty,sum(Stages_crossed) as total_stages_crossed from level_Details2
where Dev_ID like 'zm_%'
and level_ = 2
group by Difficulty
order by (total_stages_crossed)
desc;
```

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

Difficulty	total_stages_crossed
Difficult	46
Medium	35
Low	15

Q4) EXTRACT P_ID AND THE TOTAL NUMBER OF UNIQUE DATES FOR THOSE PLAYERS WHO HAVE PLAYED GAMES ON MULTIPLE DAYS

```sql
select P_ID, count(distinct(start_datetime)) as no_unique_dates
from level_Details2
group by P_ID
having no_unique_dates > 1;
```

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

P_ID	no_unique_dates
211	6
224	4
242	2
292	2
296	2
300	5
310	3
358	2
368	4
429	4
482	5

Q5) FIND P_ID AND LEVEL WISE SUM OF KILL_COUNTS WHERE KILL_COUNT IS GREATER THAN AVG KILL COUNT FOR THE MEDIUM DIFFICULTY

```sql
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_;
```

rid | Filter Rows: | Export: | Wrap Cell Content:

P_ID	Level_	total_kill_count
211	1	55
211	0	20
224	2	58
224	1	54
242	1	58

Q6) FIND LEVEL AND ITS CORRESPONDING LEVEL CODE WISE SUM OF LIVES EARNED EXCLUDING LEVEL 0. ARRANGE IN ASECENDING ORDER OF LEVEL

```
select level_Details2.Level_, player_details.L1_Code, player_details.L2_Code,
sum(level_Details2.Lives_Earned) as total_lives_earned
from level_Details2
join player_details on level_Details2.P_ID = player_details.P_ID
where Level_ <> 0
group by level_Details2.Level_,player_details.L1_Code,player_details.L2_Code
order by Level_
asc;
```

Filter Rows: | Export: | Wrap Cell Content:

Level_	L1_Code	L2_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
1	speed_blitz	cosmic_vision	4
1	speed_blitz	splippery_slope	3
1	war_zone		4
1	war_zone	resurgence	0
1	war_zone	splippery_slope	7

Q7) FIND TOP 3 SCORE BASED ON EACH DEV_ID AND RANK THEM IN INCREASING ORDER USING ROW_NUMBER. DISPLAY DIFFICULTY AS WELL

```sql
WITH ScoreRanking AS (
    SELECT
        Dev_ID, Difficulty, score,
        ROW_NUMBER() OVER (
            PARTITION BY Dev_ID
            ORDER BY score DESC
        ) AS Ranked
    FROM level_Details2
)
SELECT
    Dev_ID, Difficulty, score,Ranked
FROM ScoreRanking
WHERE Ranked <= 3
ORDER BY dev_id,Ranked ;
```

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

Dev_ID	Difficulty	score	Ranked
bd_013	Difficult	5300	1
bd_013	Difficult	4570	2
bd_013	Difficult	3370	3
bd_015	Difficult	5300	1

Q8) FIND FIRST_LOGIN DATETIME FOR EACH DEVICE ID

```sql
select Dev_ID,min(start_datetime) as first_login_datetime
from level_Details2
group by Dev_ID;
```

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

Dev_ID	first_login_datetime
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

Q9) FIND TOP 5 SCORE BASED ON EACH DIFFICULTY LEVEL AND RANK THEM IN INCREASING ORDER USING RANK. DISPLAY DEV_ID AS WELL

```sql
with RR_rank as(
select Dev_ID, Difficulty,score,
rank() over(
partition by Difficulty
order by score asc) as _Rank
from level_Details2)
select Dev_ID, Difficulty,score,  _Rank
from RR_rank
where _Rank <=5
order by Difficulty,_Rank;
```

Filter Rows: | Export: | Wrap Cell Content:

Dev_ID	Difficulty	score	_Rank
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

Q10) 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

```sql
select P_ID, Dev_ID, min(start_datetime) as first_login_date
from level_Details2
group by Dev_ID,P_ID;
```

Filter Rows: | Export: | Wrap Cell Content:

P_ID	Dev_ID	first_login_date
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

Q11) FOR EACH PLAYER AND DATE, HOW MANY KILL_COUNT PLAYED SO FAR BY THE PLAYER. THAT IS, THE TOTAL NUMBER OF GAMES PLAYED BY THE PLAYER UNTIL THAT DATE.

A) WINDOW FUNCTION

```
select P_ID,start_datetime,
sum(Kill_Count) over
(partition by P_ID, start_datetime order by start_datetime) as kills
from level_details2
order by  P_ID, start_datetime;
```

Filter Rows: | Export: | Wrap Cell Content:

P_ID	start_datetime	kills
211	2022-10-12 13:23:45	20
211	2022-10-12 18:30:30	25
211	2022-10-13 05:36:15	30
211	2022-10-13 22:30:18	14
211	2022-10-14 08:56:24	9
211	2022-10-15 11:41:19	15
224	2022-10-14 01:15:56	20
224	2022-10-14 08:21:49	34
224	2022-10-15 05:30:28	30
224	2022-10-15 13:43:50	28
242	2022-10-13 01:14:29	21

Q11) FOR EACH PLAYER AND DATE, HOW MANY KILL_COUNT PLAYED SO FAR BY THE PLAYER. THAT IS, THE TOTAL NUMBER OF GAMES PLAYED BY THE PLAYER UNTIL THAT DATE.

B) WITHOUT WINDOW FUNCTION

```sql
select P_ID,start_datetime,
sum(Kill_Count) as kills
from level_details2
group by P_ID, start_datetime
order by  P_ID, start_datetime;
```

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

P_ID	start_datetime	kills
211	2022-10-12 13:23:45	20
211	2022-10-12 18:30:30	25
211	2022-10-13 05:36:15	30
211	2022-10-13 22:30:18	14
211	2022-10-14 08:56:24	9
211	2022-10-15 11:41:19	15
224	2022-10-14 01:15:56	20

Q12) FIND THE CUMULATIVE SUM OF STAGES CROSSED OVER A START_DATETIME

```sql
select P_ID,start_datetime,
sum(Stages_crossed) over
 (partition by P_ID order by start_datetime) as stages_crossed
 from level_details2
order by  P_ID, start_datetime;
```

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

P_ID	start_datetime	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

Q13) FIND THE CUMULATIVE SUM OF AN STAGES CROSSED OVER A START_DATETIME FOR EACH PLAYER ID BUT EXCLUDE THE MOST RECENT START_DATETIME

```
with recent_start_datetime as
(select P_ID, max(start_datetime)  as recent_time
from  level_details2
group by P_ID ),
cumulative_sum as(
select  a.P_ID,
a.start_datetime,
sum(a.Stages_crossed) over(
 partition by a.P_ID order by a.start_datetime
 )  as cumulative_stages_crossed
 from
 level_details2 as a
 join recent_start_datetime rsd on a.P_ID = rsd.P_ID
 and a.start_datetime < rsd.recent_time
 )
 select P_ID, start_datetime, cumulative_stages_crossed
 from cumulative_sum
order by  P_ID,start_datetime;
```

rid | Filter Rows: | Export: | Wrap Cell Content:

P_ID	start_datetime	cumulative_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

Q14) EXTRACT TOP 3 HIGHEST SUM OF SCORE FOR EACH DEVICE ID AND THE CORRESPONDING PLAYER_ID

```sql
with high_score as
(select P_ID, Dev_ID,
sum(Score) as total_score,
 row_number() over (partition by Dev_ID order by sum(Score) desc)
 as highest_score
from level_details2
group by P_ID, Dev_ID
)
select P_ID, Dev_ID, total_score
from high_score
where highest_score <=3
order by  Dev_ID,total_score desc;
```

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

Q15) FIND PLAYERS WHO SCORED MORE THAN 50% OF THE AVG SCORE, SCORED BY SUM OF SCORES FOR EACH PLAYER_ID

```sql
select P_ID,
SUM(Score) AS total_score
from level_details2
group by P_ID
having total_score > 0.5 * (select avg(Score) from level_details2
);
```

id | Filter Rows: | Export: | Wrap Cell Content:

P_ID	total_score
211	10940
224	16310
242	6310
292	2560

Q16) CREATE A STORED PROCEDURE TO FIND TOP N HEADSHOTS_COUNT BASED ON EACH DEV_ID AND RANK THEM IN INCREASING ORDER USING ROW_NUMBER. DISPLAY DIFFICULTY AS WELL

```
Delimiter //
create procedure get_top_n_headshots(
in N int)
begin
    with top_n as (
    select
        sum(Headshots_Count) as top_n_headshots , Dev_ID, Difficulty,
    row_number() over (partition by Dev_ID order by Headshots_Count asc
        ) as n_rank
from level_details2
group by
    Dev_ID, Difficulty,Headshots_Count)
select Dev_ID, Difficulty, top_n_headshots, n_rank
from top_n
where n_rank <= N
order by
    Dev_ID, n_rank ;
    end//

call get_top_n_headshots(3);
```

Filter Rows: Export: Wrap Cell Content:

Dev_ID	Difficulty	top_n_headshots	n_rank
bd_013	Medium	4	1
bd_013	Medium	8	2
bd_013	Medium	10	3

Q17) CREATE A FUNCTION TO RETURN SUM OF SCORE FOR A GIVEN PLAYER_ID

```
DELIMITER //
create function player_score(player_id int)
returns int
DETERMINISTIC NO SQL READS SQL DATA
begin
    declare Sum_of_score int;
select
    sum(Score) into Sum_of_score
from
    level_details2
    where P_ID = player_id;
return Sum_of_score;
end//

SELECT player_score(211);
```

Filter Rows: | Export: | Wrap Cell Content:

player_score(211)
10940

THANK YOU!