SQL Round

Duration: 40 Minutes
Difficulty: Hard
Domain: Risk

Problem

An integrity team in Twitch, a video streaming platform for games, ensures that publishers follow a community guidelines that video content is not sexual, hateful or spammy. Use the violations table below to address three part questions.

Date	User_ID	Video	Sexual	Hateful	Spam
2020-01-01	ABC	1	1	1	0
2020-01-01	ABC	2	1	0	0
2020-01-01	ABC	3	0	0	0
2020-01-01	XYZ	1	0	0	0
...	...	...	...	...	...
2020-07-01	ABC	4	0	1	0
2020-07-01	BCD	4	0	0	0
2020-07-01	CDA	2	0	0	1

#1 - On a monthly basis, how many users publish at least one video that violates all three categories - sexual, hateful and spammy?

#2 - Currently, the integrity team doesn't enforce banning a user unless the number of violations exceeds ten. A revision is proposed such that a user is banned if the number of violations accumulated exceeds three. For each user, return two records:

1. The first record shows the date, user_id and status "0" when a user published a video for the first time.
2. The last record shows the date, user_id and status "1" when a user published a video for the last time before being banned.

For users who are not banned, only return the first record.

Files

> **Data File:** twitch_content_violations.csv
> **Solution File:** twitch_content_violations.txt

Solution

> **#1 -** On a monthly basis, how many users publish at least one video that violates all three categories - sexual, hateful and spammy?

[Candidate] In the violations table, I see tha some users post videos multiple times a day. Can I assume that a user could post multiple videos that violate all the discretions?

[Interviewer] Yes, but you need to ensure that you don't repeat count users who publish such videos multiple times.

[Candidate] Then is it correct to assume that if a user publishes two videos that violated all three categories, the user is counted once, not twice?

[Interviewer] Yes, that's correct.

[Candidate] Understood. Here's my query.

```
SELECT year, month, user_id, COUNT(*)
FROM (
        SELECT EXTRACT(YEAR FROM date) as year,
                EXTRACT(MONTH FROM date) as month,
                user_id
        FROM violations
        WHERE (sexual + hateful + spam) == 3
) t
GROUP BY year, month, user_id;
```

[Candidate] First**,** the subquery filters on videos with all three violations and creates new columns that indicate the year and month. The subquery table is aggregated at year, month, user to return counts of videos that violate all policies per month.

Interviewer Feedback: The candidate produced a wrong query. The query generates the count of videos that violate policies per year, month and user. It is slightly concerning that the candidate missed on a basic question. This is a slight concern. Here's the correct solution.

```
# The top-level query applies group by on year, month and counts rows that represent
# instances when users published videos that violated three rules.
SELECT year, month, COUNT(*)
FROM (
        # The sub-query first filters on videos that violate all categories then removes
        # duplicate user_id's using DISTINCT on year,month,user_id
        SELECT DISTINCT EXTRACT(YEAR FROM date) AS year,
                        EXTRACT(MONTH FROM date) AS month,
                        user_id
        FROM violations
        WHERE (sexual + hateful + spam) = 3
) t
GROUP BY year, month;
```

> **#2 -** Currently, the integrity team doesn't enforce banning a user unless the number of violations exceeds ten. A revision is proposed such that a user is banned if the number of violations accumulated exceeds three.
>
> For each user, return two records
> 1. the first record shows the date, user_id and status "0" when a user published a video for the first time.
> 2. the last record shows the date, user_id and status "1" when a user published a video for the last time before being banned.
>
> For users who are not banned, only return the first record.

[Candidate] If a user publishes a video with three flags, then would the user be kicked off the platform given the new enforcement?

[Interviewer] Not yet, the user's next video needs to contain one or more violations before banned.

[Candidate] I see. If the user posts one video on 03-01-2020 with three violations and another one on 03-02-2020 with two violations, then the cumulative number of violations that the committed is five. Therefore, the number of days the banned user was on the platform is 1 day.

[Interviewer] Yes. Also consider that a user could be banned on the same day if they publish multiple videos that break rules.

[Candidate] I have one final question for clarification. Suppose you have the following example of a user:

Date	User_ID	Video	Sexual	Hateful	Spam
2020-01-01	ABC	1	1	1	0
2020-01-01	ABC	2	1	0	0
2020-01-03	ABC	3	0	0	0
2020-01-04	ABC	4	1	0	0

Should the final table return rows with the following for the user?

Date	User_ID	Status
2020-01-01	ABC	0
2020-01-04	ABC	1

[Interviewer] Precisely.

[Candidate] Understood. Here's my initial solution:

```
SELECT user_id,
        date,
        ROW_NUMBER() OVER(PARTITION BY user_id ORDER BY date ASC) -1 AS status
FROM (
    SELECT *,
          ROW_NUMBER() OVER(PARTITION BY user_id ORDER BY video ASC) AS ascending_order,
          ROW_NUMBER() OVER(PARTITION BY user_id ORDER BY video DESC) AS descending_order
    FROM (
        SELECT *,
                SUM(total_violations) OVER (PARTITION BY user_id ORDER BY date, video ASC) AS
                                                                                cumulative_violations
        FROM (
                SELECT user_id,
                        date,
                        video,
                        (sexual + hateful + spam) AS total_violations
                FROM violations
        ) t
    ) t2
    WHERE cumulative_violations <= 4
) t3
WHERE ascending_order = 1 OR descending_order = 1;
```

[Interviewer] Can you elaborate your solution?

[Candidate] Absolutely, the first subquery "t" counts the number of violations per video. The second subquery "t2" gets the cumulative count of violations per user. The third subquery "t3" assigns 1 to the first and last record with videos that have cumulative violations less than or equal to 4. The final query isolates the first and last records and adds the status column using ROW_NUMBER().

Interviewer Feedback: The candidate wrote a solution that is incorrect and convoluted. He assumed that a user's last video is when the forth violation is committed. However, this is not always the case. A user could publish be banned on their fifth or sixth violations. A user could publish a video that violates three rules then, then publish a second video that violates more than one rule. Candidate's current query ignores such cases.Lastly, candidate's query is convoluted. The query can be cleaned using the "WITH" clause.

Here's the correct solution:

```
# The first WITH clause calculates the cumulative violations per user across dates.
WITH cumulative_violation_table AS (
SELECT user_id,
          date,
          SUM(total_violations) OVER (PARTITION BY user_id ORDER BY date, video ASC)
                                                                    AS cumulative_violations
FROM (
        SELECT user_id,
                  video,
                  date,
                  (sexual + hateful + spam) AS total_violations
        FROM violations ) t1
),
# The second WITH clause takes the sub-query that creates the status indicator below and
# sets the key dates, first and last videos, based on the date of the first video in each
# status per user.
status_minimum_date AS (
SELECT user_id,
          status,
          MIN(date) AS date
FROM (
        # The subquery creates a status indicator if the user is banned "1" or
        # not "0" on a particular day.
        SELECT user_id,
                  date,
                  CASE WHEN cumulative_violations <= 3 THEN 0 ELSE  1 END AS status
        FROM cumulative_violation_table ) t
        GROUP BY user_id, status
)
# Final table that retrieves status and date per user.
SELECT * FROM status_minimum_date ORDER BY user_id, status;
```

Final Assessment

In the SQL section, the candidate is assessed based on correctness, SQL efficiency, and communication. For each dimension the candidate is rated in the following scale: (5) superior, (4) good, (3) adequate, (2) marginal, (1) not competent.

Assessments	Rating	Comments
SQL Correctness	**1**	The candidate made sloppy mistakes in both SQL questions. In the first problem, the candidate failed to create a query that calculates the monthly count of users who publish videos flagged with all violations. In the second problem, the candidate returns the wrong end date of a user. One suggestion is to double-check his solution before finalizing it. This will ensure that the solution addresses the question asked.
SQL Efficiency	**1**	The candidate's second solution was convoluted with redundancies using partitions and ordering which are expensive operations. Had the candidate structured the SQL approach differently, the query could be simplified and efficient.
Communication	**4**	The candidate communication was good. He posed questions to strive for clarity but failed to deliver with the correct solution. The candidate must ensure that he comprehends the problem and asks more questions when required to deliver the correct solution. In addition, the candidate explained his solution quite well, but he could include more details. For instance, in each subquery he wrote, illustrate how the table is manipulated from one step to the next.