SQL Round

Format: Onsite
Duration: 40 Minutes
Difficulty: Hard
Domain: Analytics

Problem

A social network platform allows businesses to publish advertisements. The platform tracks daily advertisment spendings across business accounts. Currently, the platform doesn't have a backend system to alert unsual spendings. Address the SQL questions below:

Ads Spending Table

Date	Business	Spending
2020-01-01	ABC	10
2020-01-01	BCD	20
2020-01-01	CDA	30
2020-01-01	XYZ	20
...	...	...
2020-07-01	ABC	10
2020-07-01	BCD	20
2020-07-01	CDA	30

#1 - Compute the 30-day moving average of advertisement spending per business.

#2 - Compute the 30-day moving standard deviation of advertisement spending per business.

#3 - The platform wants to track anomalous spendings. Create a new column called "outlier" which flags any spending that is above or below the two standard deviation from the mean. Use the moving average and standard deviation computed in previous steps.

Files

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

Solution

> **#1 -** Compute the 30-day moving average of advertisement spending per business.

[Candidate] To get the moving average of the advertisement spending per business, I can add row number per business in the ascending order of dates. Selecting the first thirty rows per business then averaging the spending per business offer the solution:

```
SELECT business,
          AVG(spending) AS moving_avg
FROM (
        SELECT business,
                  ROW_NUMBER() OVER(PARTITION BY business) AS row_number
        FROM ads_spending
        WHERE row_number <= 30
) t
GROUP BY business;
```

[Interviewer] If you filter based on row_number, you wouldn't be calculating the moving average right?

[Candidate] Let me see if I can revise my query. I could use the lag function to average the past 30 day spending per business.

```
SELECT business,
          AVG(spending, lag) AS moving_avg
FROM (
        SELECT business,
                  spending,
                  LAG(spending, 30) AS lag
        FROM ads_spending
) t
GROUP BY business;
```

Interviewer Feedback: The candidate has displayed poor SQL skills. Both attempts provide incorrect solutions with logical and syntatical errors. In the first attempt, the candidate presumed that selecting the first 30 rows then calculating the moving average of spending leads to the correct solution. However, his solution merely provides the average within the first thirty dates of spending per business.

When offered a pointer to help correct the solution, the candidate once again wrote an incorrect solution with a flawed logic and syntax error. The "**LAG(**spending, 30)" will merely shift the index of the spending column by 30 rows. Secondly, the "**AVG(**spending, lag)" will produce SQL error as only one column can be computed in a statistics function.

```
SELECT *,
        # Compute the moving average of spending
        (SELECT AVG(s2.spending) FROM ads_spending s2
            WHERE (s1.date - s2.date) <= 30 AND
                  (s1.date - s2.date) >= 0 AND
                   s1.business = s2.business) AS moving_avg
FROM ads_spending s1;
```

> **#2 -** Compute the 30-day moving standard deviation of advertisement spending per business.

[Candidate] I will apply a similar approach as in the previous problem.

```
SELECT business,
          POWER(AVG(spending, lag) - spending, 0.5) AS moving_std
FROM (
        SELECT business,
                  spending,
                  LAG(spending, 30) AS lag
        FROM ads_spending
) t
GROUP BY business;
```

Interviewer Feedback: Once again, the candidate produced an incorrect solution. The candidate needs to brush up on SQL syntax and the formula for standard deviation.

```
SELECT *,
          # Compute the moving std of spending
          (SELECT POWER(AVG(POWER(s3.spending - t1.moving_avg, 2)), 0.5)
           FROM ads_spending s3
           WHERE (t1.date - s3.date) <= 30 AND
                    (t1.date - s3.date) >= 0 AND
                    t1.business = s3.business) AS moving_std
FROM (
        SELECT *,
                  # Compute the moving average of spending
                  (SELECT AVG(s2.spending) FROM ads_spending s2
                    WHERE (s1.date - s2.date) <= 30 AND
                              (s1.date - s2.date) >= 0 AND
                               s1.business = s2.business) AS moving_avg
        FROM ads_spending s1
) t1;
```

> **#3 -** The platform wants to track anomalous spendings. Create a new column called “outlier” which flags any spending that is above or below the two standard deviation from the mean. Use the moving average and standard deviation computed in previous steps.

[Candidate] To approach this problem, I could use the solutions in the first two problems. I will create a sub-query that generates standard deviation and average. Finally, I will select rows when the spending is two standard deviations from the mean.

```
SELECT *
FROM (
        SELECT business,
                POWER(AVG(spending, lag) - spending, 0.5) AS moving_avg,
                AVG(spending, lag) AS moving_std
        FROM (
                SELECT business,
                        spending,
                        LAG(spending, 30) AS lag
                FROM ads_spending
        ) t
        GROUP BY business
) t2
WHERE spending > moving_avg + 2 * moving_std AND
          spending < moving_avg - 2 * moving_std;
```

Interviewer Feedback: Candidate's provided an incorrect solution. The sub-queries will fail to run on incorrect syntax. In addition, the candidate poorly understood the objective. Instead of appending a column with flags, the candidate attempted to return just the rows that contained spendings outside bounds.

```
SELECT *,
          # Append a new column called 'outlier' that flags outlier spendings
          (CASE WHEN spending > moving_avg + 2 * moving_std THEN 1
                    WHEN spending < moving_avg - 2 * moving_std THEN 1
                    ELSE 0
           END) AS outlier
FROM (
        SELECT *,
                  # Compute the moving standard deviation
                  (SELECT POWER(AVG(POWER(s3.spending - t1.moving_avg, 2)), 0.5)
                   FROM ads_spending s3
                   WHERE (t1.date - s3.date) <= 30 AND
                              (t1.date - s3.date) >= 0 AND
                              t1.business = s3.business) AS moving_std
        FROM (
                SELECT *,
                          # Compute the moving average
                          (SELECT AVG(s2.spending) FROM ads_spending s2
                           WHERE (s1.date - s2.date) <= 30 AND
                                      (s1.date - s2.date) >= 0 AND
                                       s1.business = s2.business) AS moving_avg
                FROM ads_spending s1
        ) t1
) t2;
```

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	In all three problems, the candidate produced incorrect solutions containing syntax and logic errors. The candidate needs to brush up on basic SQL syntax such as window function that calculates statistics. He also needs to review sub-queries. For instance, in the first problem, he created a new column and applied the WHERE clause on the column within the same sub-query. Such syntax will produce an error. Lastly, he also used an incorrect formula for the standard deviation in the second problem.
SQL Efficiency	1	The candidate created solutions with flawed logic with unnecessary sub-queries.
Communication	1	The candidate poorly understood problems. For instance, in the last problem, he presumed that the final query should return only the rows that contained outliers. However, the instruction requests for a new column indicating an outlier. The candidate should consider asking questions before rushing to the solution. In addition, the candidate should walkthrough the solution with the interviewer to double-check his approach.