# Identify High-Value Transactions

## Business Objective :

The objective is to identify unusually high-value transactions for a user — which may indicate fraud, spikes in spending, or system glitches.

## Understanding OF Data:

The dataset includes over 65K rows. The adv\_id is the user ID, value\_in\_paise is the transaction amount, and created\_at gives the time sequence needed to evaluate past transactions.

#### Highlight:

* Checked for missing adv\_ids.
* Data types like timestamps were used properly.
* Focused only on valid and complete rows for accurate analysis.

## Logical Approach:

To find high-value anomalies, I needed to compare each transaction with the average of the last 3 transactions for that same user. For that, I used window functions to first rank transactions by time and then calculate the rolling average.

## Key SQL Techniques:

* ROW\_NUMBER() or RANK() to order transactions per user.
* JOIN to compare with previous rows (self-join approach).
* AVG() OVER () or explicit windowing to compute the rolling average.

I filtered for cases where the transaction is at least twice the rolling average of the last three, skipping users with fewer than 3 past transactions.

## Performance Concerns:

This approach can be slow on very large datasets. So indexing on adv\_id and created\_at, or using native LAG() / AVG() OVER functions where supported, would improve performance.

## Case Wrap Up:

This logic can power fraud detection alerts, user spending dashboards, or anomaly detection pipelines, and is reusable across apps or currencies.

## Query:

WITH ranked\_txn AS (

SELECT

adv\_id,

created\_at,

value\_in\_paise,

ROW\_NUMBER() OVER (PARTITION BY adv\_id ORDER BY created\_at) AS txn\_rank

FROM transactions

WHERE adv\_id IS NOT NULL

),

rolling\_avg AS (

SELECT

t1.adv\_id,

t1.created\_at,

t1.value\_in\_paise,

AVG(t2.value\_in\_paise) AS avg\_last\_3\_txns

FROM ranked\_txn t1

JOIN ranked\_txn t2

ON t1.adv\_id = t2.adv\_id

AND t2.txn\_rank BETWEEN t1.txn\_rank - 3 AND t1.txn\_rank - 1

GROUP BY t1.adv\_id, t1.created\_at, t1.value\_in\_paise

)

SELECT \*

FROM rolling\_avg

WHERE avg\_last\_3\_txns IS NOT NULL

AND value\_in\_paise >= 2 \* avg\_last\_3\_txns

ORDER BY adv\_id, created\_at;