

SQL Data Analyst Project

- Synthetic Financial Dataset for Fraud Detection



BY Sangini Phansekar

1. Detecting Recursive Fraudulent Transactions

Question:

Use a recursive CTE to identify potential money laundering chains where money is transferred from one account to another across multiple steps, with all transactions flagged as fraudulent.

Solution:

This query uses a recursive CTE to track the flow of money through multiple accounts over successive steps. The recursive part of the CTE allows us to follow the chain of transactions and identify patterns that could indicate money laundering activities. It filters out chains where all transactions are marked as fraudulent.

```
WITH RECURSIVE fraud_chain as (  
  SELECT nameOrig as initial_account,  
         nameDest as next_account,  
         step,  
         amount,  
         newbalanceorig  
  FROM  
    transactions  
  WHERE isFraud = 1 and type = 'TRANSFER'  
  
  UNION ALL  
  
  SELECT fc.initial_account,  
         t.nameDest,t.step,t.amount ,t.newbalanceorig  
  FROM fraud_chain fc  
  JOIN transactions t  
  ON fc.next_account = t.nameorig and fc.step < t.step  
  where t.isfraud = 1 and t.type = 'TRANSFER')  
  
SELECT * FROM fraud_chain
```

2. Analyzing Fraudulent Activity over Time

Question:

Use a CTE to calculate the rolling sum of fraudulent transactions for each account over the last 5 steps.

Solution : This query uses a CTE to calculate the cumulative sum of fraudulent transactions for each account over the last five steps. It helps in understanding the temporal distribution of fraudulent activities, which is crucial for identifying patterns over time.

```
with rolling_fraud as ( SELECT nameorig,step,  
    SUM(isfraud) OVER (PARTITION BY nameOrig order by STEP ROWS BETWEEN 4 PRECEDING and CURRENT ROW ) as fraud_rolling  
FROM transactions)  
  
SELECT * FROM rolling_fraud  
WHERE fraud_rolling > 0
```

3. Complex Fraud Detection Using Multiple CTEs

Question:

Use multiple CTEs to identify accounts with suspicious activity, including large transfers, consecutive transactions without balance change, and flagged transactions.

```
WITH large_transfers as (  
  SELECT nameOrig,step,amount FROM transactions WHERE type = 'TRANSFER' and amount >500000),  
no_balance_change as (  
  SELECT nameOrig,step,oldbalanceOrig,newbalanceOrig FROM transactions where oldbalanceOrig=newbalanceOrig),  
flagged_transactions as (  
  SELECT nameOrig,step FROM transactions where isflaggedfraud = 1)  
  
SELECT  
  lt.nameOrig  
FROM  
  large_transfers lt  
JOIN  
  no_balance_change nbc ON lt.nameOrig = nbc.nameOrig AND lt.step = nbc.step  
JOIN  
  flagged_transactions ft ON lt.nameOrig = ft.nameOrig AND lt.step = ft.step;
```

4. Write me a query that checks if the computed `new_updated_Balance` is the same as the actual `newbalanceDest` in the table. If they are equal, it returns those rows.

```
with CTE as (  
  SELECT amount,nameorig,oldbalancedest,newbalanceDest,(amount+oldbalancedest) as new_updated_Balance  
  FROM transactions  
)  
SELECT * FROM CTE where new_updated_Balance = newbalanceDest;
```

5. Detect Transactions with Zero Balance Before or After

- **Question:** Find transactions where the destination account had a zero balance before or after the transaction.
- **SQL Prompt:** Write a query to list transactions where `oldbalanceDest` or `newbalanceDest` is zero.

```
SELECT nameOrig, nameDest, amount, oldbalanceDest, newbalanceDest
FROM transactions
WHERE oldbalanceDest = 0 OR newbalanceDest = 0 limit 10;
```