SQL Data Analyst Project

By Chirag Kapoor

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 RecursiveFraudChain AS (
    -- Anchor: Starting fraud transfers
    SELECT nameOrig AS initial_account,
           nameDest AS next_account,
           step,
           amount,
           newbalanceOrig
    FROM [dbo].[Transaction]
    WHERE isFraud = 1 AND type = 'TRANSFER'

    UNION ALL

    -- Recursive: Chain next fraud transfers
    SELECT fc.initial_account,
           t.nameDest AS next_account,
           t.step,
           t.amount,
           t.newbalanceOrig
    FROM RecursiveFraudChain fc
    JOIN [dbo].[Transaction] t ON fc.next_account = t.nameOrig
                              AND fc.step < t.step
    WHERE t.isFraud = 1 AND t.type = 'TRANSFER'
)
SELECT * FROM RecursiveFraudChain
OPTION (MAXRECURSION 100);
```

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 [dbo].[Transaction]
)
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.

```sql
WITH large_transfers AS (
    SELECT
        nameOrig,
        step,
        amount
    FROM [dbo].[Transaction]
    WHERE type = 'TRANSFER'
      AND amount > 500000
),
no_balance_change AS (
    SELECT
        nameOrig,
        step,
        oldbalanceOrg,
        newbalanceOrig
    FROM [dbo].[Transaction]
    WHERE oldbalanceOrg = newbalanceOrig
),
flagged_transactions AS (
    SELECT
        nameOrig,
        step
    FROM [dbo].[Transaction]
    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 [dbo].[Transaction]
    )
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.

```
With XTE as (
    Select amount, nameOrig, oldbalanceDest, newbalanceDest
    From [dbo].[Transaction]
    )
Select * From XTE where oldbalanceDest = 0 or newbalanceDest = 0;
```