FINANCIAL DATASET FOR FRAUD DETECTION

QUESTIONS

1.DETECTING RECURSIVE FRAUDULENT TRANSACTIONS

2. ANALYZING FRAUD ACTIVITY OVER TIME

3. COMPLEX FRAUD DETECTION USING MULTIPLE CITIES

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.

1.DETECTING RECURSIVE FRAUDULENT TRANSACTIONS

```sql
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 FRAUD ACTIVITY OVER TIME

THIS QUERY USES A CTE TO CALCUATE THE CUMULATIVE SUM OF FRAUDENT TRANSACTION FOR EACH ACCOUNT OVER THE LAST FIVE STEPS

```
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 CITIES.

```sql
WITH large_transfers as (
SELECT nameOrig,step,amount FROM transactions WHERE type = 'TRANSFER' and amount >500000),
no_balance_change as (
SELECT nameOrig,step,oldbalanceOrg,newbalanceOrig FROM transactions where oldbalanceOrg=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.

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

IN THIS PROJECT I HAVE CREATED AND UPLOADED THE DATA THROUGH MYSQL 8.0 COMMAND LINE

```
    ->     isFlaggedFraud TINYINT
    -> );
ERROR 1046 (3D000): No database selected
mysql> use bank
Database changed
mysql> CREATE TABLE transactions (
    ->     step INT,
    ->     type VARCHAR(20),
    ->     amount DECIMAL(15,2),
    ->     nameOrig VARCHAR(20),
    ->     oldbalanceOrg DECIMAL(15,2),
    ->     newbalanceOrig DECIMAL(15,2),
    ->     nameDest VARCHAR(20),
    ->     oldbalanceDest DECIMAL(15,2),
    ->     newbalanceDest DECIMAL(15,2),
    ->     isFraud TINYINT,
    ->     isFlaggedFraud TINYINT
    -> );
Query OK, 0 rows affected (0.07 sec)

mysql> LOAD DATA INFILE "D:/MYSQL/Bank data set/PS_20174392719_1491204439457_log.csv"
    -> INTO TABLE transactions
    -> FIELDS TERMINATED BY ','
    -> ENCLOSED BY '"'
    -> LINES TERMINATED BY '\n'
    -> IGNORE 1 ROWS;
Query OK, 6362620 rows affected (3 min 4.79 sec)
Records: 6362620  Deleted: 0  Skipped: 0  Warnings: 0

mysql> SELECT * FROM transactions li
```