Forensic Data Analytics System

Goal: Build a fraud detection database to uncover anomalies in financial transactions.

Tables:

- `accounts(account_id, customer_id, balance, opened_date)`
- `transactions(transaction_id, account_id, amount, transaction_type, timestamp)`

Queries:

- Detects accounts with unusually high transaction frequency.
- Flag transactions above a certain threshold compared to the customer's average.

a. Create database

```
CREATE DATABASE forensic_analytics;
USE forensic_analytics;
```

```
mysql> CREATE DATABASE forensic_analytics;
Query OK, 1 row affected (0.03 sec)

mysql> USE forensic_analytics;
Database changed
```

b. Create tables

```
-- Customer accounts
CREATE TABLE accounts (
    account_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT,
    balance DECIMAL(12,2),
    opened_date DATE
);

-- Transactions
CREATE TABLE transactions (
    transaction_id INT AUTO_INCREMENT PRIMARY KEY,
    account_id INT,
    amount DECIMAL(12,2),
    transaction_type VARCHAR(50), -- e.g., 'Deposit', 'Withdrawal',
'Transfer'
    timestamp DATETIME,
    FOREIGN KEY (account_id) REFERENCES accounts(account_id)
);
```

```
mysql> -- Customer accounts
mysql> CREATE TABLE accounts (
    ->     account_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     customer_id INT,
    ->     balance DECIMAL(12,2),
    ->     opened_date DATE
    -> );
Query OK, 0 rows affected (0.10 sec)

mysql>
mysql> -- Transactions
mysql> CREATE TABLE transactions (
    ->     transaction_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     account_id INT,
    ->     amount DECIMAL(12,2),
    ->     transaction_type VARCHAR(50), -- e.g., 'Deposit', 'Withdrawal', 'Transfer'
    ->     timestamp DATETIME,
    ->     FOREIGN KEY (account_id) REFERENCES accounts(account_id)
    -> );
Query OK, 0 rows affected (0.12 sec)
```

c. Insert sample data

```
INSERT INTO accounts (customer_id, balance, opened_date)
VALUES
(101, 5000.00, '2024-01-15'),
(102, 12000.00, '2024-03-10'),
(103, 800.00, '2024-05-20');

INSERT INTO transactions (account_id, amount, transaction_type,
timestamp)
VALUES
(1, 2000.00, 'Deposit', '2025-10-01 09:00:00'),
(1, 4500.00, 'Withdrawal', '2025-10-02 14:00:00'),
(2, 10000.00, 'Transfer', '2025-10-03 11:30:00'),
(3, 700.00, 'Withdrawal', '2025-10-04 16:00:00'),
(3, 50.00, 'Deposit', '2025-10-05 10:00:00');
```

```
mysql> INSERT INTO accounts (customer_id, balance, opened_date)
    -> VALUES
    -> (101, 5000.00, '2024-01-15'),
    -> (102, 12000.00, '2024-03-10'),
    -> (103, 800.00, '2024-05-20');
Query OK, 3 rows affected (0.04 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
mysql> INSERT INTO transactions (account_id, amount, transaction_type, timestamp)
    -> VALUES
    -> (1, 2000.00, 'Deposit', '2025-10-01 09:00:00'),
    -> (1, 4500.00, 'Withdrawal', '2025-10-02 14:00:00'),
    -> (2, 10000.00, 'Transfer', '2025-10-03 11:30:00'),
    -> (3, 700.00, 'Withdrawal', '2025-10-04 16:00:00'),
    -> (3, 50.00, 'Deposit', '2025-10-05 10:00:00');
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

d. Example queries

```
-- Detect accounts with unusually high transaction frequency
SELECT account_id, COUNT(*) AS transaction_count
FROM transactions
GROUP BY account_id
HAVING transaction_count > 3;

-- Flag transactions above a threshold compared to customer's
```

```
average
SELECT t.account_id, t.transaction_id, t.amount,
       AVG(t2.amount) AS avg_amount
FROM transactions t
JOIN transactions t2 ON t.account_id = t2.account_id
GROUP BY t.account_id, t.transaction_id
HAVING t.amount > 2 * avg_amount;

-- Find suspicious withdrawals (e.g., > 80% of account balance)
SELECT a.account_id, t.transaction_id, t.amount, a.balance
FROM accounts a
JOIN transactions t ON a.account_id = t.account_id
WHERE t.transaction_type = 'Withdrawal'
  AND t.amount > (0.8 * a.balance);
```

```
mysql> -- Detect accounts with unusually high transaction frequency
mysql> SELECT account_id, COUNT(*) AS transaction_count
    -> FROM transactions
    -> GROUP BY account_id
    -> HAVING transaction_count > 3;
Empty set (0.04 sec)

mysql>
mysql> -- Flag transactions above a threshold compared to customer's average
mysql> SELECT t.account_id, t.transaction_id, t.amount,
    ->        AVG(t2.amount) AS avg_amount
    -> FROM transactions t
    -> JOIN transactions t2 ON t.account_id = t2.account_id
    -> GROUP BY t.account_id, t.transaction_id
    -> HAVING t.amount > 2 * avg_amount;
Empty set (0.01 sec)

mysql>
mysql> -- Find suspicious withdrawals (e.g., > 80% of account balance)
mysql> SELECT a.account_id, t.transaction_id, t.amount, a.balance
    -> FROM accounts a
    -> JOIN transactions t ON a.account_id = t.account_id
    -> WHERE t.transaction_type = 'Withdrawal'
    ->   AND t.amount > (0.8 * a.balance);
+------------+----------------+---------+---------+
| account_id | transaction_id | amount  | balance |
+------------+----------------+---------+---------+
|          1 |              2 | 4500.00 | 5000.00 |
|          3 |              4 |  700.00 |  800.00 |
+------------+----------------+---------+---------+
2 rows in set (0.01 sec)
```