Banking Transactions & Customer Insights

Goal: Build a secure financial transactions database with anonymization.

Tables:

- `customers(customer_id, anonymized_id, region, age_group)`
- `transactions(transaction_id, anonymized_id, amount, transaction_type, timestamp)`

Queries:

- Aggregate spending by anonymized customer groups.
- Detects unusual transaction spikes across regions.

a. Create database

```
CREATE DATABASE banking_insights;
USE banking_insights;
```

```
mysql> CREATE DATABASE banking_insights;
Query OK, 1 row affected (0.05 sec)

mysql> USE banking_insights;
Database changed
```

b. Create tables

```
-- Customer information with anonymization
CREATE TABLE customers (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    anonymized_id VARCHAR(20) UNIQUE,  -- UNIQUE ensures foreign
key works
    region VARCHAR(50),
    age_group VARCHAR(20)
);

-- Transactions linked to anonymized customers
CREATE TABLE transactions (
    transaction_id INT AUTO_INCREMENT PRIMARY KEY,
    anonymized_id VARCHAR(20),
    amount DECIMAL(12,2),
    transaction_type VARCHAR(50), -- e.g., 'Deposit', 'Withdrawal',
'Transfer'
    timestamp DATETIME,
    FOREIGN KEY (anonymized_id) REFERENCES customers(anonymized_id)
);
```

```
mysql> CREATE TABLE customers (
    ->     customer_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     anonymized_id VARCHAR(20) UNIQUE, -- add UNIQUE here
    ->     region VARCHAR(50),
    ->     age_group VARCHAR(20)
    -> );
Query OK, 0 rows affected (0.15 sec)

mysql>
mysql> CREATE TABLE transactions (
    ->     transaction_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     anonymized_id VARCHAR(20),
    ->     amount DECIMAL(12,2),
    ->     transaction_type VARCHAR(50),
    ->     timestamp DATETIME,
    ->     FOREIGN KEY (anonymized_id) REFERENCES customers(anonymized_id)
    -> );
Query OK, 0 rows affected (0.10 sec)
```

c. Insert sample data

```
INSERT INTO customers (anonymized_id, region, age_group)
VALUES
('CUST001', 'Jakarta', '20-29'),
('CUST002', 'Bandung', '30-39'),
('CUST003', 'Surabaya', '40-49');

INSERT INTO transactions (anonymized_id, amount, transaction_type,
timestamp)
VALUES
('CUST001', 200.00, 'Deposit', '2025-09-01 09:00:00'),
('CUST001', 150.00, 'Withdrawal', '2025-09-02 10:00:00'),
('CUST002', 500.00, 'Transfer', '2025-09-03 14:30:00'),
('CUST003', 1200.00, 'Deposit', '2025-09-04 11:00:00'),
('CUST003', 800.00, 'Withdrawal', '2025-09-05 16:00:00');
```

```
mysql> INSERT INTO customers (anonymized_id, region, age_group)
    -> VALUES
    -> ('CUST001', 'Jakarta', '20-29'),
    -> ('CUST002', 'Bandung', '30-39'),
    -> ('CUST003', 'Surabaya', '40-49');
Query OK, 3 rows affected (0.04 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
mysql> INSERT INTO transactions (anonymized_id, amount, transaction_type, timestamp)
    -> VALUES
    -> ('CUST001', 200.00, 'Deposit', '2025-09-01 09:00:00'),
    -> ('CUST001', 150.00, 'Withdrawal', '2025-09-02 10:00:00'),
    -> ('CUST002', 500.00, 'Transfer', '2025-09-03 14:30:00'),
    -> ('CUST003', 1200.00, 'Deposit', '2025-09-04 11:00:00'),
    -> ('CUST003', 800.00, 'Withdrawal', '2025-09-05 16:00:00');
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

d. Example queries

```
-- Aggregate spending by age group
SELECT age_group, SUM(amount) AS total_spending
FROM customers c
JOIN transactions t ON c.anonymized_id = t.anonymized_id
GROUP BY age_group;
```

```sql
-- Detect unusual transaction spikes across regions
SELECT region, AVG(amount) AS avg_amount, MAX(amount) AS max_amount
FROM customers c
JOIN transactions t ON c.anonymized_id = t.anonymized_id
GROUP BY region;

-- Find most common transaction types
SELECT transaction_type, COUNT(*) AS frequency
FROM transactions
GROUP BY transaction_type
ORDER BY frequency DESC;
```

```
mysql> -- Aggregate spending by age group
mysql> SELECT age_group, SUM(amount) AS total_spending
    -> FROM customers c
    -> JOIN transactions t ON c.anonymized_id = t.anonymized_id
    -> GROUP BY age_group;
+-----------+----------------+
| age_group | total_spending |
+-----------+----------------+
| 20-29     |         350.00 |
| 30-39     |         500.00 |
| 40-49     |        2000.00 |
+-----------+----------------+
3 rows in set (0.00 sec)

mysql>
mysql> -- Detect unusual transaction spikes across regions
mysql> SELECT region, AVG(amount) AS avg_amount, MAX(amount) AS max_amount
    -> FROM customers c
    -> JOIN transactions t ON c.anonymized_id = t.anonymized_id
    -> GROUP BY region;
+----------+-------------+------------+
| region   | avg_amount  | max_amount |
+----------+-------------+------------+
| Jakarta  |  175.000000 |     200.00 |
| Bandung  |  500.000000 |     500.00 |
| Surabaya | 1000.000000 |    1200.00 |
+----------+-------------+------------+
3 rows in set (0.03 sec)

mysql>
mysql> -- Find most common transaction types
mysql> SELECT transaction_type, COUNT(*) AS frequency
    -> FROM transactions
    -> GROUP BY transaction_type
    -> ORDER BY frequency DESC;
+------------------+-----------+
| transaction_type | frequency |
+------------------+-----------+
| Deposit          |         2 |
| Withdrawal       |         2 |
| Transfer         |         1 |
+------------------+-----------+
3 rows in set (0.00 sec)
```