Venmito Data Investigation: Transfer-Transaction Relationships

Investigation Summary

While analyzing the Venmito database, I discovered an interesting relationship between transfers and transactions that wasn't explicitly modeled in the system. Some transfers appear to represent payments for subscription services, with amounts matching transaction subtotals from specific stores.

This document outlines the investigation process and includes query snippets that can be demonstrated live during discussion.

Key Finding

The most clear example is user "Fey Kuser" (ID 998), who received 55 transfers of exactly $20.00 from another account. These transfers match a transaction at "Sparkmart" for subscription services named "Kittykat" and "Flixnet".

Investigation Queries

1. Identify Frequent Transfer Recipients

This query identifies accounts that frequently receive transfers, which might indicate service providers:

```sql
SELECT
    p.user_id,
    p.first_name,
    p.last_name,
    p.email,
    COUNT(*) as times_received,
    SUM(t.amount) as total_received,
    COUNT(DISTINCT t.sender_id) as unique_senders
FROM transfers t
JOIN people p ON t.recipient_id = p.user_id
GROUP BY p.user_id, p.first_name, p.last_name, p.email
HAVING COUNT(*) > 10  -- Look for accounts that received more than 10 transfers
ORDER BY times_received DESC, total_received DESC
LIMIT 20;
```

2. Find Matching Amounts Between Transfers and Transactions

This query identifies common monetary values that appear in both transfers and transactions:

sql

```sql
WITH common_transfer_amounts AS (
    SELECT
        ROUND(amount, 2) as rounded_amount,
        COUNT(*) as transfer_count
    FROM transfers
    GROUP BY ROUND(amount, 2)
    HAVING COUNT(*) >= 5
),
transaction_subtotals AS (
    SELECT
        t.transaction_id,
        t.store,
        ROUND(SUM(ti.subtotal), 2) as total_subtotal,
        COUNT(*) as transaction_count
    FROM transactions t
    JOIN transaction_items ti ON t.transaction_id = ti.transaction_id
    GROUP BY t.transaction_id, t.store
)
SELECT
    cta.rounded_amount,
    cta.transfer_count,
    COUNT(DISTINCT ts.total_subtotal) as matching_transaction_count,
    array_agg(DISTINCT ts.store) as stores
FROM common_transfer_amounts cta
JOIN transaction_subtotals ts ON cta.rounded_amount = ts.total_subtotal
GROUP BY cta.rounded_amount, cta.transfer_count
ORDER BY cta.transfer_count DESC
LIMIT 20;
```

3. Key Discovery: Temporal Relationship Between Transfers and Transactions

This crucial query reveals transfers occurring close in time to transactions with matching amounts:

sql

```sql
WITH transaction_subtotals AS (
    -- Calculate total subtotals for each transaction
    SELECT
        t.transaction_id,
        t.user_id as buyer_id,
        t.store,
        t.transaction_date,
        SUM(ti.subtotal) as calculated_total,
        t.price as recorded_total
    FROM transactions t
    JOIN transaction_items ti ON t.transaction_id = ti.transaction_id
    GROUP BY t.transaction_id, t.user_id, t.store, t.transaction_date, t.price
)
SELECT
    tr.transfer_id,
    tr.sender_id,
    p_sender.first_name || ' ' || p_sender.last_name AS sender_name,
    tr.recipient_id,
    p_recipient.first_name || ' ' || p_recipient.last_name AS recipient_name,
    tr.amount as transfer_amount,
    tr.timestamp as transfer_time,
    ts.transaction_id,
    ts.store,
    ts.calculated_total,
    ts.transaction_date,
    ABS(EXTRACT(EPOCH FROM (tr.timestamp - ts.transaction_date)) / 86400) as days_difference
FROM transfers tr
JOIN transaction_subtotals ts ON ABS(ROUND(tr.amount, 2) - ROUND(ts.calculated_total, 2)) < 0.0
JOIN people p_sender ON tr.sender_id = p_sender.user_id
JOIN people p_recipient ON tr.recipient_id = p_recipient.user_id
WHERE ABS(EXTRACT(EPOCH FROM (tr.timestamp - ts.transaction_date)) / 86400) < 7  -- Within 7 da
ORDER BY days_difference, ABS(tr.amount - ts.calculated_total);
```

4. Examine Transaction Details for the Identified Pattern

This query shows the specific items in the transaction that matches our discovered transfer pattern:

sql

```sql
-- Examine the Sparkmart transaction in detail
SELECT
    t.transaction_id,
    t.user_id,
    t.store,
    t.price,
    t.transaction_date,
    ti.item,
    ti.quantity,
    ti.price_per_item,
    ti.subtotal
FROM transactions t
JOIN transaction_items ti ON t.transaction_id = ti.transaction_id
WHERE t.transaction_id = 'T034';  -- The transaction ID from our findings
```

5. Identify Recurring Transfer Patterns Between Users

This query finds recurring transfer patterns that might indicate subscription payments:

sql

```sql
WITH sender_recipient_pairs AS (
    SELECT
        sender_id,
        recipient_id,
        amount,
        COUNT(*) as transfer_count,
        COUNT(DISTINCT ROUND(amount, 2)) as unique_amounts,
        ROUND(COUNT(*) / COUNT(DISTINCT ROUND(amount, 2)), 2) as transfers_per_unique_amount
    FROM transfers
    GROUP BY sender_id, recipient_id, amount
    HAVING COUNT(*) >= 3  -- At least 3 transfers with the same amount
)
SELECT
    sp.sender_id,
    sp.recipient_id,
    ps.first_name || ' ' || ps.last_name as sender_name,
    pr.first_name || ' ' || pr.last_name as recipient_name,
    pr.email as recipient_email,
    sp.amount,
    sp.transfer_count,
    sp.unique_amounts,
    sp.transfers_per_unique_amount
FROM sender_recipient_pairs sp
JOIN people ps ON sp.sender_id = ps.user_id
JOIN people pr ON sp.recipient_id = pr.user_id
ORDER BY sp.transfer_count DESC, sp.transfers_per_unique_amount DESC
LIMIT 20;
```

6. Analyze Transfer Patterns for Potential Subscription Accounts

This query identifies accounts receiving regular transfers of the same amount—a pattern consistent with subscription payments:

sql

```sql
SELECT
    p.user_id,
    p.first_name || ' ' || p.last_name AS name,
    p.email,
    COUNT(*) as transfer_count,
    COUNT(DISTINCT sender_id) as unique_senders,
    ROUND(COUNT(*) / COUNT(DISTINCT sender_id), 2) as transfers_per_sender,
    COUNT(DISTINCT amount) as unique_amounts,
    ROUND(COUNT(*) / COUNT(DISTINCT amount), 2) as transfers_per_amount,
    array_agg(DISTINCT amount) as common_amounts
FROM transfers t
JOIN people p ON t.recipient_id = p.user_id
GROUP BY p.user_id, p.first_name, p.last_name, p.email
HAVING COUNT(*) >= 5 AND -- At least 5 transfers
       COUNT(*) / COUNT(DISTINCT amount) >= 2 -- Each amount appears multiple times
ORDER BY transfers_per_amount DESC, transfer_count DESC
LIMIT 10;
```

Research Findings

1. **Subscription Payments**: Users appear to be making regular transfers that match transaction amounts for subscription services.
2. **Hidden Store/Service Accounts**: Some user accounts function as service providers, receiving regular payments via transfers.
3. **Temporal Correlation**: Transfers often occur within days of corresponding transactions, suggesting a causal relationship.
4. **Example Discovery**:
   - User "Fey Kuser" (ID 998) received 55 transfers of $20.00 from another "Fey Kuser" account (ID 1002)
   - These match a $20.00 Sparkmart transaction for "Kittykat" and "Flixnet" services
   - Transfers occur as close as 1 day from the transaction date

Potential Applications of This Discovery

1. **Enhanced Data Model**:
   - Flag subscription service accounts in the system
   - Create explicit links between transfers and corresponding transactions
   - Classify transfer types (e.g., "subscription payment")

2. **User Experience Improvements**:
    - Better labeling of subscription transfers
    - Subscription management interface
    - Spending analytics for subscription services
3. **Business Intelligence**:
    - Track popular subscription services
    - Monitor subscription retention
    - Identify payment patterns and user behaviors