# **Transaction Reconciliation System - Take-Home Coding Exercise**

## **Overview**

You are tasked with building a transaction reconciliation system that matches transaction records with settlement data and provides a reporting dashboard to identify potential issues.

## **Business Context**

Financial transactions go through a settlement process where the actual money movement occurs. Your system needs to:

1. Match transactions with their corresponding settlements
2. Identify discrepancies and potential issues
3. Provide a clear dashboard for financial operations teams

## **Technical Requirements**

### **Data Sources Provided**

1. **SQLite Database** (reconciliation.db) containing:
   * transactions table
   * settlement\_history table
2. **CSV file** (settlement\_report.csv) with new settlement data to reconcile

### **Database Schema**

#### **transactions table**

****CREATE TABLE transactions (

transaction\_id VARCHAR(50) PRIMARY KEY,

lifecycle\_id VARCHAR(50),

account\_id VARCHAR(50) NOT NULL,

merchant\_name VARCHAR(100) NOT NULL,

transaction\_date DATE NOT NULL,

transaction\_amount DECIMAL(10,2) NOT NULL,

currency VARCHAR(3) NOT NULL,

status VARCHAR(20) NOT NULL,

settlement\_status VARCHAR(20) DEFAULT 'PENDING',

total\_settled\_amount DECIMAL(10,2) DEFAULT 0.00,

last\_settlement\_date DATE,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

#### **settlement\_history table**

****CREATE TABLE settlement\_history (

settlement\_id VARCHAR(50) PRIMARY KEY,

transaction\_id VARCHAR(50),

lifecycle\_id VARCHAR(50),

settlement\_date DATE NOT NULL,

settlement\_amount DECIMAL(10,2) NOT NULL,

settlement\_type VARCHAR(10) NOT NULL, -- 'DEBIT' or 'CREDIT'

currency VARCHAR(3) NOT NULL,

processed\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (transaction\_id) REFERENCES transactions(transaction\_id)

);

#### **Sample transactions data**

### ****INSERT INTO transactions (transaction\_id, lifecycle\_id, account\_id, merchant\_name, transaction\_date, transaction\_amount, currency, status, settlement\_status, total\_settled\_amount, last\_settlement\_date) VALUES

### -- Completed transactions pending settlement

### ('TXN001', 'LC001', 'ACC123', 'Amazon.com', '2025-08-14', 125.99, 'USD', 'COMPLETED', 'PENDING', 0.00, NULL),

### ('TXN002', 'LC002', 'ACC124', 'Starbucks', '2025-08-14', 4.75, 'USD', 'COMPLETED', 'PENDING', 0.00, NULL),

### ('TXN003', 'LC003', 'ACC125', 'Shell Gas', '2025-08-14', 45.30, 'USD', 'COMPLETED', 'PENDING', 0.00, NULL),

### ('TXN004', 'LC004', 'ACC126', 'Best Buy', '2025-08-14', 899.99, 'USD', 'COMPLETED', 'PENDING', 0.00, NULL),

### ('TXN005', 'LC005', 'ACC127', 'Home Depot', '2025-08-14', 234.50, 'USD', 'COMPLETED', 'PENDING', 0.00, NULL),

### ('TXN006', 'LC006', 'ACC128', 'Target', '2025-08-14', 87.25, 'USD', 'COMPLETED', 'PENDING', 0.00, NULL),

### ('TXN007', 'LC007', 'ACC129', 'Walmart', '2025-08-14', 156.78, 'USD', 'COMPLETED', 'PENDING', 0.00, NULL),

### ('TXN008', 'LC008', 'ACC130', 'Apple', '2025-08-05', 1299.99, 'USD', 'COMPLETED', 'PENDING', 0.00, NULL),

### ('TXN009', 'LC009', 'ACC131', 'Nike', '2025-08-14', 189.95, 'USD', 'COMPLETED', 'PENDING', 0.00, NULL),

### ('TXN010', 'LC010', 'ACC132', 'Netflix', '2025-08-07', 15.99, 'USD', 'COMPLETED', 'PENDING', 0.00, NULL),

### ('TXN013', 'LC011', 'ACC135', 'Corner Deli', '2025-08-14', 32.45, 'USD', 'COMPLETED', 'PENDING', 0.00, NULL),

### ('TXN014', 'LC012', 'ACC136', 'Shell Gas', '2025-08-14', 28.90, 'USD', 'COMPLETED', 'PENDING', 0.00, NULL),

### -- Failed transactions (should not have settlements)

### ('TXN015', 'LC013', 'ACC137', 'Amazon.com', '2025-08-14', 199.99, 'USD', 'FAILED', 'NOT\_APPLICABLE', 0.00, NULL),

### ('TXN016', 'LC014', 'ACC138', 'Uber Taxi', '2025-08-14', 75.25, 'USD', 'DECLINED', 'NOT\_APPLICABLE', 0.00, NULL);

### 

### **CSV Settlement Report Format**

****settlement\_id,lifecycle\_id,account\_id,merchant\_name,transaction\_date,settlement\_date,settlement\_amount,settlement\_type,currency

SET001,LC001,ACC123,Amazon.com,2025-08-14,2025-08-15,125.99,DEBIT,USD

SET002,LC002,ACC124,Starbucks,2025-08-14,2025-08-15,4.75,DEBIT,USD

SET003,LC003,ACC125,Shell Gas,2025-08-14,2025-08-15,35.30,DEBIT,USD

SET004,LC004,ACC126,Best Buy,2025-08-14,2025-08-15,899.99,DEBIT,USD

SET005,LC004,ACC126,Best Buy,2025-08-14,2025-08-15,20.00,CREDIT,USD

SET006,LC005,ACC127,Home Depot,2025-08-14,2025-08-15,150.00,DEBIT,USD

SET007,LC005,ACC127,Home Depot,2025-08-14,2025-08-15,84.50,DEBIT,USD

SET008,LC006,ACC128,Target,2025-08-14,2025-08-15,87.25,DEBIT,USD

SET009,LC007,ACC129,Walmart,2025-08-14,2025-08-15,180.78,DEBIT,USD

SET010,LC008,ACC130,Apple,2025-08-05,2025-08-06,1299.99,DEBIT,USD

SET011,LC008,ACC130,Apple,2025-08-05,2025-08-08,299.99,CREDIT,USD

SET012,LC009,ACC131,Nike,2025-08-14,2025-08-15,189.95,DEBIT,USD

SET014,,ACC135,Corner Deli,2025-08-14,2025-08-15,32.45,DEBIT,USD

SET015,,ACC136,Shell Gas,2025-08-14,2025-08-15,28.90,DEBIT,USD

SET016,LC999,ACC199,Disney Plus,2025-08-14,2025-08-15,99.99,DEBIT,USD

SET017,,ACC137,Amazon.com,2025-08-14,2025-08-15,199.99,DEBIT,USD

SET018,,ACC777,Uber Taxi,2025-08-10,2025-08-11,25.75,DEBIT,USD

SET019,LC002,ACC124,Starbucks,2025-08-14,2025-08-15,1.50,CREDIT,USD

SET020,LC006,ACC128,Target,2025-08-14,2025-08-15,12.25,CREDIT,USD

**Note**: Some records in the CSV may not have lifecycle\_id. In such cases, match using account\_id, merchant\_name, and transaction\_date.

## **Functional Requirements**

### **1. Data Processing & Reconciliation**

* Load and parse the CSV settlement report
* Implement matching logic with fallback strategy:
  + Primary: Match by lifecycle\_id
  + Fallback: Match by account\_id + merchant\_name + transaction\_date
* Update the settlement\_history table with new settlement records
* **Update transaction settlement status** based on reconciliation results:
  + PENDING - No settlements yet
  + PARTIAL - Settled amount < transaction amount
  + FULLY\_SETTLED - Settled amount = transaction amount
  + OVER\_SETTLED - Settled amount > transaction amount
  + REFUNDED - Net settlement after credits < transaction amount
* Calculate and update total\_settled\_amount (net of debits and credits)
* Update last\_settlement\_date with the most recent settlement
* Handle multiple settlements per transaction
* Support both DEBIT and CREDIT settlement types

### **2. Issue Detection Algorithm**

Identify and flag transactions with the following issues:

**Critical Issues (Red Flag):**

* Total settlement amount > transaction amount
* No settlement after 7 days from transaction date

**Warning Issues (Yellow Flag):**

* Total settlement amount < transaction amount

### **3. Dashboard Requirements**

#### **Main Dashboard View**

* **Reconciliation Summary Panel**:
  + Total transactions processed
  + Total settlements processed
  + **Breakdown by settlement status** (Pending, Partial, Fully Settled, Over Settled)
  + Number of transactions with critical issues
  + Number of transactions with warnings
  + Total outstanding amount (unreconciled)
  + **Settlement performance metrics** (avg days to settle, settlement rate)

#### **Transaction List View**

* Display all transactions in a table
* Show columns: Transaction ID, Date, Merchant, Amount, **Settlement Status**, Settlement Amount, Issue Flag
* **Filter by settlement status** (Pending, Partial, Fully Settled, etc.)
* Visual indicators for issue types (red for critical, yellow for warnings)
* **Settlement status badges** with color coding
* Click on any transaction to view details

#### **Transaction Detail View**

* Transaction information
* Settlement history table for that specific transaction
* Issue details and recommendations
* Settlement timeline visualization (bonus points)

## **Technical Specifications**

### **Backend Requirements**

* **Language**: Choose from [Node.js](http://node.js) (Typescript), Python, Golang, or Kotlin
* **Database**: SQLite (provided)
* **API**: RESTful endpoints for:
  + POST /reconcile - Process CSV and reconcile settlements
  + GET /transactions - List all transactions with issue flags
  + GET /transactions/{id} - Get transaction details with settlements
  + GET /dashboard/summary - Get reconciliation summary stats

### **Frontend Requirements**

* **Framework**: React, Vue.js, or vanilla JavaScript
* **Responsive design** for desktop and tablet
* **Data visualization**: Charts for summary statistics
* **User experience**: Intuitive navigation and clear issue indication

### **Data Validation**

* Validate CSV format and data types
* Handle missing or malformed data gracefully
* Ensure currency consistency
* Validate settlement amounts (positive values)

## **Deliverables**

### **1. Code Repository**

* Well-structured codebase with clear separation of concerns
* README with setup and run instructions
* Requirements/dependencies file

### **2. Database Setup**

* SQL scripts to create and populate test data
* Data migration/seeding scripts

### **3. Documentation**

* **Architecture Overview**: Brief explanation of your solution approach
* **API Documentation**: Endpoint specifications
* **Assumptions Made**: Any business logic assumptions
* **Known Limitations**: Areas for future improvement

### **4. Testing**

* Unit tests for core reconciliation logic
* Integration tests for API endpoints
* Test data examples

## **Sample Test Scenarios**

### **Scenario 1: Perfect Match**

* Transaction: $100.00
* Settlement: $100.00 (DEBIT)
* Expected: No issues

### **Scenario 2: Partial Settlement**

* Transaction: $100.00
* Settlement: $80.00 (DEBIT)
* Expected: Warning flag

### **Scenario 3: Over Settlement**

* Transaction: $100.00
* Settlements: $100.00 (DEBIT) + $20.00 (CREDIT - refund)
* Net Settlement: $80.00
* Expected: No issues

### **Scenario 4: Over Settlement Critical**

* Transaction: $100.00
* Settlement: $120.00 (DEBIT)
* Expected: Critical issue

### **Scenario 5: Missing Lifecycle ID**

* CSV record without lifecycle\_id but matching account\_id, merchant\_name, transaction\_date
* Expected: Successful matching via fallback logic

### **Scenario 6: Status Transitions**

* Transaction: $100.00 (initially PENDING)
* First Settlement: $50.00 (DEBIT) → Status: PARTIAL
* Second Settlement: $50.00 (DEBIT) → Status: FULLY\_SETTLED
* Expected: Proper status transitions and total\_settled\_amount updates

### **Scenario 7: Complex Refund Case**

* Transaction: $100.00
* Settlements: $100.00 (DEBIT) + $30.00 (CREDIT - partial refund)
* Net Settlement: $70.00, Status: REFUNDED
* Expected: Correct net calculation and status assignment

## **Getting Started**

1. Download the provided SQLite database and CSV sample files
2. Set up your development environment
3. Implement the reconciliation logic first
4. Build the API layer
5. Create the dashboard interface
6. Write tests and documentation

## **Bonus Features (Optional)**

* Export functionality (CSV/PDF reports)
* Advanced filtering and search capabilities on the dashboard

## **Submission Instructions**

1. Create a Git repository with your solution
2. Include a comprehensive README with setup instructions
3. Provide a brief video demo (2-3 minutes) of your working solution
4. Share the repository link and any additional notes

## 

## **Questions?**

If you have any questions about the requirements or need clarification on business rules, please don't hesitate to reach out. We encourage asking questions as it demonstrates thoughtful analysis of the problem.

In addition, feel free to make reasonable assumptions about requirements not explicitly specified. Document your assumptions in the README.