Unified Account-to-Account "Ultimate Table"

Background

Your company's fraud detection models for account-to-account payments need comprehensive features to identify sophisticated fraud patterns like money mule networks and authorised push payment fraud. Currently, payment-relevant data is scattered across multiple systems, making it difficult for data scientists to create the rich feature set needed for effective fraud detection.

Current Data Landscape

- **Payment Transaction System**: Real-time A2A payment data (amount, timestamp, account IDs, payment reference)
- **Account Database**: Account details, opening dates, customer demographics, account status (daily batch updates)
- **Historical Fraud Cases**: Confirmed fraud labels, investigation outcomes, recovery amounts (manual updates with 30-60 day delay)
- **External Risk Feeds**: Known fraudulent account numbers, suspicious sort codes (irregular updates from industry sharing) Data Integration

Challenges

- Historical fraud labels come in different formats according to the submitting institution. Part of them come from an API, via individual label submission. The rest is sent in via files (csv, excel), often manually compiled by bank analysts.
- External risk feeds have varying quality and update frequencies.

Task One: Data Consolidation Strategy

Design a strategy to create and maintain a unified payment intelligence table that provides a comprehensive view of all the available data while accounting for the complexity of multiple data sources (as specified in the Current Data Landscape) and timing constraints.

Deliverables

- **Data Architecture Design** (2-5 pages): Real-time serving, batch processing, and feature computation strategy .
- **Data Quality & Monitoring Strategy**: Ensuring data reliability and consistency with upstream data flows.

Task Two: Data Consolidation Pipeline

Objective:

Build a Python script that consolidates multiple data sources into a single unified table representing account-to-account payments.

For this task, you have **four data sources**:

Payment Transactions
- Columns:

```
payment_id, src_account_id, dest_account_id, payment_reference, amount, timestamp
```

Account Details
- Columns:

```
account_id, opening_date
```

External Risk Feed
- Columns:

```
account_id, risk_flag (1=suspicious, 2=confirmed)
```

Historical Fraud Cases
- Columns:

```
payment_id, fraud_type, fraud_reported_date
```

Requirements

- Read these four datasets (assume they are provided as CSV files).
- Merge them into **one consolidated table** with the following columns:
  ```
  payment_id | src_account_id | dest_account_id | payment reference | amount | timestamp |
  src_opening_date | dest_opening_date | src_risk_flag | dest_risk_flag | fraud_flag (boolean) |
  fraud_type | fraud_reported_date
  ```
- Ensure:
  - Source and destination account details are correctly joined.
  - Missing risk flags default to `0=False`.
  - Missing fraud details (`fraud_type`, `fraud_reported_date`) default to `None` or empty.
- Output the final table as a CSV file named `unified_table.csv`.

Constraints

- Use **Python**.
- Code should be **clear, modular, and runnable**.

Bonus Challenge!

The following is a stretch task: it is not strictly necessary to pass this round of the interview.
If coding the whole thing is too much, even just discussing your approach is acceptable.

Assume the following changes to the data ingestion process:

- **Payment Transactions** come from a **live API stream** (simulate with a generator or mock API calls).
- **Account Details** are updated **daily via CSV**.
- **External Risk Feed** arrives in **irregular batch loads** (simulate by loading multiple files at different times).
- **Historical Fraud Cases** come from another **live API stream**, but with a **delay relative to the original transaction** (simulate by introducing a lag in fraud data availability).

For this challenge:

- Design your code so it can **handle these different ingestion patterns**.
- You can simulate streaming by reading data in chunks or using Python generators.
- Document your approach clearly.