Got it ✅ — here’s a **hands-on introduction for your automation expert** so they can pick up the sample file, understand its structure and logic, and start **reverse-engineering the ETL → Postgres set-up** with confidence.

# **📘 Introduction to the Reconciliation Sample File**

*(for Automation Engineer — ETL + Postgres POC)*

## **1. Purpose of the File**

This Excel workbook is the **single source of truth** showing:

1. **Inputs** — 4 sheets that represent raw data the business provides each month.
2. **Outputs** — 3 sheets that represent the reconciliations the business currently performs manually in Excel.

👉 Your job: **load the inputs into Postgres, replicate the same outputs with SQL views, and validate parity.**

## **2. Input Sheets (the raw feeds)**

These are the four “CSVs” you will eventually automate. For now, treat each sheet as one **raw table** in raw.\*.

### **(a) External Accounts (Merchant)**

* Represents **pull requests** made to collect funds from merchant bank accounts.
* Key fields:  
  + Beneficiary Bank Account Number → merchant’s VA number (join key).
  + Buy Amount (SGD) → amount pulled.
  + Created Date → posting date.

👉 **Business meaning:** “What we *tried* to pull from the merchant.”

### **(b) VA Transaction Report (all)**

* Ledger of **all inflows and outflows** per Virtual Account (VA).
* Key fields:  
  + Transaction Date
  + Receiver Virtual Account Number
  + Sender Virtual Account Number
  + Remarks (critical — tells you what the transaction is: merchant-repayment, fh-admin-fee, sr-interest, etc.)
  + Amount

👉 **Business meaning:** “What actually happened in the VA ledger (money in/out).”  
 👉 **ETL note:** This is your **fact table** of truth for all reconciliations.

### **(c) Repmt-SKU (by Note)**

* Expected repayments for each SKU/Note.
* Columns include expected **fees, principal, interest** (senior, junior, admin, etc.).

👉 **Business meaning:** “What we *expected* to pay out.”  
 👉 **ETL note:** Will need **unpivoting** (wide → tall format) to align with categories in VA remarks.

### **(d) Repmt-Sales Proceeds (by Note)**

* UI/admin record of inflows.
* Columns: SKU ID, Total Funds Inflow, Sales Proceeds.

👉 **Business meaning:** “What the UI says came in.”  
 👉 **ETL note:** Used for Level 1 + Level 2b checks.

## **3. Output Sheets (what we must replicate in SQL views)**

### **(1) Formula & Output → Level 1**

* Columns:  
  + SKU ID, Account Number (VA), Merchant
  + Amount Pulled (from External Accounts)
  + Amount Received (from VA ledger, inflows where remark = 'merchant-repayment')
  + Sales Proceeds (from Repmt-Sales Proceeds)
  + Variances (Pulled vs Received, Pulled vs Sales)

👉 **SQL View:** mart.v\_level1

### **(2a) Formula & Output → Level 2a**

* Waterfall breakdown: Paid vs Expected.
* Categories (from VA “remarks”):  
  + Acquirer Fees, Admin Fees, Interest, Principal (Sr/Jr), etc.
* Expected values (from Repmt-SKU).
* Paid values (from VA outflows).
* Net checks: Received inflow – roundings ≈ distributed outflows.

👉 **SQL View:** mart.v\_level2a

### **(2b) Formula & Output → Level 2b**

* UI vs VA reconciliation.
* Compares UI values (Repmt-Sales Proceeds, Repmt-SKU) vs VA-derived values.
* Flags mismatches.

👉 **SQL View:** mart.v\_level2b

## **4. Mapping Tables (needed for ETL)**

These are **not in the file** but are implied. You must build them as small reference tables (ref.\*):

1. **Note–SKU–VA Map (ref.note\_sku\_va\_map)**
   * Needed because inputs reference Note ID, SKU ID, or VA Number inconsistently.
   * Fields: note\_id, sku\_id, va\_number, merchant\_code.
   * Without this, you cannot join across sheets reliably.
2. **Remarks → Category Map (ref.remarks\_category\_map)**
   * Maps raw remarks in VA Txns to business categories used in waterfall (e.g., fh-admin-fee → FH Admin Fee).
   * Needed for Level 2a logic.

## **5. ETL → Postgres Flow**

Here’s how to structure your pipeline:

### **Step 1: Load to raw.\***

* raw.external\_accounts
* raw.va\_txn
* raw.repmt\_sku
* raw.repmt\_sales

*(Exact schemas = sheet columns; load as-is, then clean.)*

### **Step 2: Clean + type → core.\***

* Normalize dates → date / period\_ym
* Cast amounts → numeric(18,2)
* Trim/lowercase remarks
* Attach note\_sku\_va\_map for keys

### **Step 3: Reference joins → ref.\***

* Add mappings for joins and categories.

### **Step 4: Business views → mart.\***

* mart.v\_level1 → Pulled vs Received vs Sales Proceeds
* mart.v\_level2a → Waterfall Paid vs Expected
* mart.v\_level2b → UI vs VA

## **6. Reverse Engineering the Excel**

* **Start with Level 1**:  
  + Take one SKU/merchant from the Excel output.
  + Trace backwards: Amount Pulled → External Accounts, Amount Received → VA Txns, Sales Proceeds → Repmt-Sales.
  + Write a SQL query to reproduce.
* **Then move to Level 2a**:  
  + Take categories (Admin Fee, Sr Interest, etc.).
  + Look at VA outflows for that SKU (Remarks = fee type).
  + Compare to Repmt-SKU expected column.
* **Finally Level 2b**:  
  + Compare UI values vs VA sums.
  + Output mismatches.

## **7. Maintenance & Scale**

* **Schema-first:** Define Postgres schemas for raw, ref, core, mart.
* **Validation:** Run parity checks each load (SQL totals vs Excel totals).
* **Mappings:** Keep ref.\* tables versioned (Git or CSVs in /ref/).
* **Idempotency:** Key = (merchant\_code, period\_ym). Always re-runnable.
* **Performance:** Index on (merchant\_code, period\_ym, sku\_id) for all core tables.

## **8. Your First Tasks (concrete starting point)**

1. Create 4 raw.\* tables mirroring the sheets.
2. Load **1 merchant / 1 month** sample from the file.
3. Build ref.note\_sku\_va\_map (from the Excel output where both SKU + VA appear).
4. Write a SQL query to reproduce **Level 1** totals.
5. Compare with Excel — must match exactly.

✅ If you can do this for **Level 1, one merchant**, you have proven the ETL path.  
 From there, it’s just expanding to more merchants, adding Level 2a/2b logic, and then connecting the DB to the agent for OPS-friendly insights.

Would you like me to also prepare a **ready-to-use Postgres schema DDL script** (all 4 raw tables + reference tables + starter mart views), so your automation expert can bootstrap without guessing?