Project Title:

Unified Transaction Reconciliation Platform (UTRP) using SQL & Power BI

Data Analyst:

Shijin Ramesh | September 2025

---

1. Business Problem

Organizations often struggle with reconciling invoices vs. payments across multiple systems.

- Invoices are generated in one system, while payments may come from different banks, gateways, or ERPs.
- Mismatches occur due to partial payments, overpayments, duplicate payments, or missing payments.
- Manual reconciliation is time-consuming, error-prone, and lacks transparency.

The business needed a robust, automated reconciliation system to:

- Identify discrepancies across large datasets.
- Provide actionable insights to finance teams.
- Enable visual exception reporting for faster resolution.

---

2. Project Objective

The project aimed to design and implement an end-to-end reconciliation engine with:

- Data Integration: Merge and standardize customer, invoice, and payment data.
- Reconciliation Rules: Apply layered logic (matched, partial, overpaid, unpaid).
- Exception Tracking: Log unresolved mismatches with suggested actions.
- Analytics & Visualization: Build a Power BI dashboard to provide transparency to business users.

---

3. Scope of Work

- Phase 1: Define matching rules and acceptance criteria.
- Phase 2: Build canonical schema (fact & dimension tables) in Snowflake.

- Phase 3: Develop merge stored procedures (*sp_merge_customers*, *sp_merge_invoices*, *sp_merge_payments*) with audit logging.
- Phase 4: Implement reconciliation stored procedure (*sp_reconcile_invoices_payments*) with exception handling.
- Phase 5: Build an interactive Power BI dashboard for reconciliation KPIs and drilldown analysis.

---

4. Tools & Technologies Used

- SQL (Snowflake): Schema design, stored procedures, reconciliation logic.
- Power BI: Data modeling, DAX measures, dashboard visualizations.
- Excel/CSV: Sample dataset preparation and loading.

---

5. Implementation Phases

Phase 1: Matching Rules

- Defined reconciliation acceptance criteria:
  - **Matched**: Invoice total = Payment total.
  - **Partial**: Payments < Invoice total.
  - **Overpaid**: Payments > Invoice total.
  - **Unpaid**: No payment received.

Phase 2: Schema Design

- Built canonical tables:
  - *dim_customer* (customer master)
  - *fact_invoice* (invoice transactions)
  - *fact_payment* (payment transactions, extended with invoice_id for invoice-level matching)
  - *recon_invoice_payment* (reconciliation results)
  - *recon_exceptions* (exception log with suggested actions)
- Purpose: Create a clean single source of truth for reconciliation.

Phase 3: Merge Procedures

- Created idempotent merge stored procedures to load and update canonical tables from staging:
  - *sp_merge_customers*
  - *sp_merge_invoices*
  - sp_merge_*payments*
- Added audit logging (*audit_merge_log*) to track rows inserted/updated per run.
- Ensured data pipelines are repeatable and reliable.

Phase 4: Reconciliation Engine

- Developed *sp_reconcile_invoices_payments* procedure to:
  - Match invoices with payments.
  - Classify outcomes into *MATCHED*, *PARTIAL*, *OVERPAID*, *UNPAID*.
  - Populate *recon_invoice_payment* and *recon_exceptions*.
  - Track each run with a unique *run_id* and timestamps for auditability.
- Key Insight from Test Runs:
  - Out of 200 invoices, only a small percentage matched initially (~5–8%).
  - Majority fell into overpaid or partial, due to simulation data.
  - Demonstrated real-world finance challenge where payments often don't align perfectly.

Phase 5: Power BI Dashboard

Created an interactive dashboard with:

- **Executive KPIs**: Total Invoices, Total Payments, Match %, Total Exceptions.
- **Trend Line**: Match % across reconciliation runs.
- **Exception Explorer**: Bar chart of exceptions by issue type (Overpaid, Partial, Unpaid).
- **Customer Drilldown**: Matrix by customer showing match status.

- **Root Cause Table**: Detailed exceptions with invoice-level breakdown and suggested actions.

Key Finding: Exceptions dominated because earlier test runs accumulated, but trend logic proved that the system can track reconciliation performance over time.

---

6. Deliverables

- **SQL Scripts**: Schema creation, stored procedures, reconciliation engine.
- **Audit Logs**: Historical records of data merges and reconciliations.
- **Power BI Dashboard (.pbix):** End-to-end visual analytics.
- **Documentation**: Project proposal, matching rules, and final report.

---

7. Achievements

- Built a working SQL + Power BI reconciliation platform from scratch.
- Implemented idempotent merge logic with audit tracking.
- Designed a flexible reconciliation engine that classifies exceptions with business logic.
- Delivered a Power BI dashboard for business consumption.
- Gained hands-on experience in end-to-end data analytics workflow (ETL → Business Rules → BI).

---

8. Next Steps (Future Enhancements)

- Automate data ingestion using Snowflake Tasks and Streams.
- Deploy reconciliation as a scheduled pipeline.
- Enhance Power BI dashboard with filters by date range, system, or currency.
- Introduce predictive analytics (forecasting exception trends).

---

9. Conclusion

This project successfully demonstrated how to design and implement a financial reconciliation system using SQL and Power BI.

The solution handles data integration, reconciliation logic, exception handling, and visualization, enabling finance teams to:

- Gain transparency into mismatches,
- Prioritize resolution efforts, and
- Improve operational efficiency.

Shijin Ramesh | Data Analyst

LinkedIn | Portfolio | GitHub