Project title

Unified Transactions Reconciliation Platform (UTRP)

Data Analyst: Shijin Ramesh

Project overview

Build an SQL-driven system that unifies invoices and payments from multiple source systems, automatically reconciles transactions, and provides a Power BI dashboard for easy exception handling and month-end close visibility.

Business Problem

Many organizations run multiple systems for billing, payments, and ERP. These systems produce fragmented, inconsistent data with different keys, formats, timing, and duplicates. As a result:

- Finance and operations teams spend many hours on manual reconciliation each month.
- Month-end close is delayed and error-prone.
- There is weak auditability and no consistent exception-tracking process.
- Missed or duplicate payments reduce cash visibility.

Project goal is to design end-to-end data solutions that solve these issues using SQL and Power BI.

Objective

Create an end-to-end, repeatable solution that:

1. Canonicalizes and merges invoices and payments from different source systems into a single, auditable schema.
2. Runs automated reconciliation logic via idempotent SQL stored procedures and records exceptions.
3. Provides a Power BI reconciliation dashboard with drill-down and actionable insights for operations.
4. Improves reconciliation speed, reduces manual effort, and strengthens audit trails.

Success metrics (examples): reduce manual reconciliation time by 60%, reduce month-end exception count by 40%, and provide full traceability for every matched/unmatched transaction.

Scope of work

The project will cover:

1. **Data discovery and sample dataset**
    - Review source table samples and keys (billing, gateway payments, ERP).
    - Create synthetic test data that covers common edge cases (partial payments, duplicates, currency issues).
2. **Schema design**
    - Design a canonical schema for customers, invoices, payments, reconciliation results, and exception logs.
3. **ETL & Merge procedures**
    - Create staging (`stg_*`) tables and batch load logic.
    - Write idempotent stored procedures to merge staging data into canonical tables (handle inserts, updates, SCD-like behavior, deduplication).
    - Add audit logging table for each run (rows processed/inserted/updated/error message).
4. **Reconciliation logic**
    - Implement stored procedures to run matching rules (exact match, partial payments, aggregated matches, date windows).
    - Populate `recon` and `recon_exceptions` tables with details and resolution suggestions.
5. **Power BI dashboard**
    - Build a Reconciliation dashboard with Executive Summary, Exception Explorer, Heatmap by source, and Root Cause pages.
    - Implement DAX measures for matched amounts, outstanding, exception aging, and resolution time.
6. **Testing & validation**
    - Create unit tests and sample scenarios to validate idempotency and correctness.
    - Run data quality checks and reconciliation reconciliation QA scripts.
7. **Documentation and Handover**
    - Provide README, deployment notes, stored procedure comments, and a short demo walkthrough video (optional).

Implementation plan — Step-by-step (high level)

Phase 1: Prepare & Understand - Collect sample source schemas (or use provided synthetic files). - Define matching rules and acceptance criteria with stakeholders.

Phase 2: Schema & Staging - Create canonical tables (dim_customer, fact_invoice, fact_payment, recon tables, audit tables).

Phase 3: Merge Stored Procedures - Write `sp_merge_customers`, `sp_merge_invoices`, `sp_merge_payments` (idempotent using `MERGE` or UPSERT patterns). - Implement audit logging and error handling. - Run merge procedures against sample data and validate counts.

Phase 4: Reconciliation Engine - Implement `sp_reconcile_invoices_payments` with layered matching rules (exact, partial, aggregated). - Populate `recon_invoice_payment` and `recon_exceptions` with reasons and suggested actions. - Add stored procedures to mark exceptions as resolved.

Phase 5: Power BI Dashboard - Import canonical and recon tables into Power BI (Import mode or DirectQuery depending on volume). - Build pages: Executive Summary, Reconciliation Heatmap, Exception Explorer, Root Cause. - Add DAX measures for key KPIs and implement visuals with filters and drill-through.

Phase 6: Testing, Documentation & Delivery - Run end-to-end tests and unit-test scripts. - Prepare README, deployment notes, and a short demo video or script. - Handover artifacts: SQL scripts, `.pbix`, sample data, test cases.

Tools & Technologies used

- **Database/SQL engine:** Snowflake - stored procedures, MERGE, transactions. (Adaptable to PostgreSQL or Snowflake.)
- **BI:** Power BI Desktop and Power BI Service for dashboarding and sharing.
- **Orchestration (optional):** Azure Data Factory or SSIS for scheduling ETL jobs.
- **Versioning & docs:** Git, Markdown README, and short demo video (MP4).

Deliverables

1. **Schema and DDL scripts**: SQL file to create canonical, staging, recon, and audit tables.
2. **Stored procedures**: T-SQL files for `sp_merge_*` and `sp_reconcile_*` with comments and error handling.
3. **Sample datasets**: CSVs for customers, invoices, payments (full and small quick-test versions).
4. **SQL insert / load scripts**: ready-to-run insert statements and a single combined SQL file for quick setup.
5. **Power BI file**: `.pbix` with the reconciliation dashboard, DAX measures, and a short guide on how to refresh.
6. **Test cases & unit tests** — SQL scripts that validate idempotency and reconciliation rules and sample expected outcomes.

7. **Documentation**: README with design decisions, matching rules, how to run procedures, scaling notes, and deployment steps.
8. **Demo walkthrough**: short script or recorded clip (if requested) showing the end-to-end flow from load → reconcile → dashboard.

Success criteria

- All invoices and payments from sample data are loaded into canonical tables without duplicates.
- Reconciliation stored procedure correctly identifies matches and creates exceptions for edge cases in the test dataset.
- Power BI dashboard shows accurate KPIs and allows drilling into exceptions.
- Deliverables are clear, reproducible, and can be run by another engineer.

Assumptions & risks

Assumptions: - Using schemas generated from the sample synthetic dataset. Access to a SQL instance and Power BI Desktop for testing.

Risks: - Highly inconsistent source keys may require fuzzy matching logic (extra time). Very large volumes will need performance tuning (partitioning, batching).