A FinTech SQL Database Project

Project Objective

The core objective of this project was to design a robust and scalable database system that simulates real-world operations in a financial technology (FinTech) environment. This wasn't just about creating tables and storing data, it was about building a clean, connected system where data flows smoothly from point of entry to analysis and can power both internal operations and user-facing services.

I used a FinTech company (Sproutly Africa) as a case study to structure the project around realistic business processes such as onboarding, wallet creation, transactions, and internal fund transfers.

Project Scope and Highlights

🔹 Core Objectives:

- Create a normalized, relational database to simulate FinTech operations.

- Ensure data integrity and clean relational flow between entities.

- Use stored procedures to mimic backend business logic.

- Perform realistic user and transaction analysis using SQL queries.

- Prepare the database for future integration with analytics platforms like Power BI.

🔹 Tables and Relationships:

- Users: Stores basic user information.

- Wallets: Linked 1:1 with users; tracks balances and wallet types.

- Transactions: Logs all credits, debits, and internal transfers.

- Schools & Businesses: Optional for specialized wallet types.

- All tables are normalized to at least 3NF (Third Normal Form).

Key Stored Procedures

Name	Description
create_user_and_wallet	Onboards a new user and automatically creates a linked wallet.
process_transaction	Processes a transaction (credit or debit) and updates wallet balance accordingly.

transfer_funds	Transfers funds from one wallet to another and ensures proper balance updates for both parties.

Realistic Use Case Simulations

To ensure the database was not only well-structured but functional, I simulated realistic user activities:

1. User Onboarding – A user signs up and a wallet is created automatically.

2. Wallet Credit – Another user sends money, and the wallet is credited.

3. Internal Transfer – The user sends money to another wallet (peer-to-peer transfer).

4. Transaction Logs & Balances – Queries were used to validate that wallet balances, transaction records, and user activity data were updated correctly.

SQL Analysis Queries

To test data flow and usability, I created SQL queries grouped in phases:

- Phase 1 – General Overview: Total users, wallet distribution, average balance, etc.

- Phase 2 – Transaction Insights: Total value transferred, top transfer sources, internal transfer breakdown.

- Phase 3 – User Activity: Top users by transaction count, inactive users, engagement patterns.

Reflections

This project helped reinforce core SQL and database management concepts while stretching into real-world system modeling. From normalization to backend logic via stored procedures, it reflects the kind of structured data foundation that powers live financial applications. This also clarified the critical role a database plays not just in storing information, but in ensuring accurate, secure, and real-time responses for both internal systems and users.