Title: Maintaining Data Quality at Current: A Senior Data Scientist's Experience

### Introduction

In this blog post, we hear from [Nanyi Dong](https://www.linkedin.com/in/nanyi-dong/), until recently a senior data scientist at Current, a fintech startup in New York. [Current](https://current.com/) is a neo-bank targeting middle-income users. Unlike traditional banks, Current does not maintain any physical branch locations. Instead, their entire operation is cloud-based and they interact with customers through a mobile app.

The primary sources of revenue for Current are transaction fees from users' card swipes and money market interest income. Current’s business model emphasizes acquiring a large user base and encouraging customers to set up direct deposits for their paychecks onto the platform. These sorts of “payroll users” are valuable because they lead to higher conversion and long-term retention rates. Recurring deposits ultimately result in more card swipes and increased revenue for the company.

### Nanyi’s role at Current

Nanyi was the first data scientist at Current and joined the company when it was only 30 employees. During Nanyi's tenure, he worked on data science projects in a variety of product areas, including core banking, product optimization, fraud detection, and customer service chat analysis, among others.

Since Current operated on an embedded data scientist model, with data scientists embedded into product teams, for any given application of data science, Nanyi was responsible for managing the entire data stream. This encompassed upstream data collection and mining, data modeling for fraud detection and user retention prediction, and downstream activities like data visualization and dashboard maintenance. Moreover, Nanyi played a crucial role in resolving data trustworthiness issues that arose between different teams.

### Current's Data Stack

Current’s data stack evolved over the course of Nanyi’s tenure. Initially, the company relied primarily on event data that was directly deposited into Google BigQuery. Additionally, they maintained a separate graph database for user information.

Over the years, though, the company transitioned to a service-oriented architecture, with each service managing its own data and definitions and generating data beyond just event-based information. Backend engineers defined these new data types and were in charge of delivering raw tables into the data warehouse.

Data scientists were then responsible for developing higher-level metrics by joining tables from different services, creating user-based metrics and tables dependent on multiple raw data sources.

The company used **BigQuery** for data warehousing and **Apache Airflow** for orchestration. For transformation and testing, they used **DBT**.

### Division of Responsibilities between Engineers and Data Scientists in terms of Data Quality

At Current, the division of responsibility was drawn at the data warehouse: engineers were responsible for defining services and producing raw tables in the data warehouse, while data scientists were responsible for everything after.

This division of responsibilities sometimes led to disagreements between the two teams. For example, the engineering team, which did some amount of pre-processing on the data before producing the raw tables, always thought the raw tables were in better shape than the data scientists thought. In their mind, the raw tables could be used directly by the business.

From the data scientists’ perspective, though, the raw tables were unintuitive, as they lacked a layer designed specifically for business use. For example:

* The column names in the raw tables were fashioned in an engineering-centric manner, making them less intuitive for business users.
* The raw tables used timestamps in raw epoch milliseconds.
* The data was scattered across a large number of service tables, necessitating advanced SQL skills for a business person to answer even surface-level questions using the raw tables.
* Raw tables would contain seven or eight timestamps for a single transaction. These timestamps recorded various stages of the transaction process, such as when the transaction was initiated, when it was processed, and when it was settled.

This gap between “engineering” data and “ready-for-data-scientist consumption” data had real consequences. For instance, a common complaint from the engineering team was that data scientists often selected the wrong timestamp for transactions. Different teams required different timestamps depending on their focus. The fraud team needed the initiation timestamp, as that was when fraudulent activity typically began. The finance team, on the other hand, was more interested in the settlement timestamp, as it represented the point at which the transaction amount became irreversible. The customer service team, however, may have been more concerned with when users interacted with the system.

Ultimately, to alleviate this problem, the data scientists created data definition pages in Confluence for each table, which laid out detailed documentation and definitions for each column. They also added comments for columns in DBT. This enabled users to select the appropriate timestamps for their specific project purposes and improve overall data usage within the organization.

### Core Types of Data at Current

After the data scientists had transformed the raw tables into usable “core tables”, there ended up being really two most frequently used tables: one about transactions and the other about users.

These tables consisted of hundreds of columns and relied on sub-level tables for joining and defining data. For example, the transaction table required the classification of transactions, especially for deposits, to identify payroll users accurately. The payroll classification column was determined by another table, which matched transactions to their categories. This table was maintained by a separate pipeline.

### Core Metrics

Like at most other companies, analytics and dashboarding were top data consumers at Current. The metrics that executives were most concerned about included:

* number of payroll users
* user acquisitions
* user retention
* overall transaction volume
* revenue
* fraud issues—precisely, whether the number of fraud incidents increased or decreased.

### Data Quality Issues Encountered

Some of the main data quality issues encountered at Current included null values, freshness, and duplicates. Duplicate transaction data was particularly problematic because a single transaction could have multiple stages. When two tables with duplicate values were joined, this created an explosion of bad data that became very difficult to debug.

### Solutions to Data Quality Issues

Current adopted DBT tests and checks to address data quality issues. These checks, which included checks on the null rate, lateness, and duplication, helped reduce workload and improve data quality. After implementing these basic checks, the data was generally in good shape for further analysis and model building.

### Data for Machine Learning Models

A separate attribution collection was used for online machine learning models. The data team's daily pipeline dumps operated as batch jobs and only updated every twenty-four hours, with table column latency sometimes reaching one or two days. This setup was suitable for general business analysis purposes, but not for certain machine learning models that needed faster reaction times. These machine learning models were mainly applied to user feature decisions, for example, whether to enroll a user into light credit products like overdrafts and instant deposits.

To achieve this, the team designed a real-time collection of user attributes, such as the number of transactions in recent days, failed transactions, and total settled deposit amounts. The data was stored in BigQuery, and a streaming database was used for real-time updates. An engineering-maintained service, rather than SQL, was responsible for updating the data. The machine learning models were then built on top of these real-time attributes.