# MTN MoMo Transaction Analysis Dashboard

**Technical Report**

Table of Contents

## 1. Introduction

### Project Overview

This project processes MTN MoMo SMS transaction data into a structured database and presents it via an interactive dashboard. It serves as a tool for financial tracking and analytics.

### Objectives

- Extract and categorize transactions from XML.
- Store data in a query-optimized database.
- Visualize trends via a responsive frontend.

## 2. Methodology

### Data Processing Pipeline

- Input: Raw XML SMS data (~900 messages).
- Parsing: Python’s `ElementTree` + regex for categorization.
- Output: SQLite database with cleaned transactions.

### Database Design

| **Field** | **Type** | **Description** |
|---|---|---|
| `type` | TEXT | Transaction category (e.g., "Incoming Money") |
| `amount` | REAL | Amount in RWF |

| | | |
|---|---|---|
| date | TEXT | Formatted as YYYY-MM-DD |

**Frontend-Backend Integration**

- API: Flask serves JSON data to the frontend.
- Dashboard: Dynamic filters + Chart.js visualizations.

## 3. Implementation Challenges & Solutions

### Main Challenges faced

#### I. SMS Parsing Complexity

***Challenge*:** MTN's SMS formats varied significantly across transaction types.

***Solution*:** Implemented a multi-layer regex system with prioritized pattern matching to handle different phrasing variants.

#### II. Data Consistency

***Challenge*:** Missing or malformed data in amount fields and sender/receiver information.

***Solution*:** Established data validation rules with default values (0.0 for amounts, "Unknown" for parties) and comprehensive logging.

#### III. Performance Optimization

***Challenge*:** Rendering 900+ transactions caused frontend lag.

***Solution*:** Implemented pagination (50 items/page) and lazy-loaded visualizations.

#### Timestamp Handling

***Challenge*:** SMS dates were in non-standard Unix milliseconds format.

***Solution*:** Automated conversion to readable `YYYY-MM-DD` format using Python's `datetime` while preserving original timestamps for auditability.

### Edge Case Transactions

***Challenge*:** Some SMS messages contained unconventional phrasing that didn't match standard patterns.

***Solution*:** Created a review log for manual inspection while ensuring valid transactions processed uninterrupted.

## 4. Key Design Decisions

### Database Schema

- We chose a single-table design for simplicity.
- Trade-off: Less normalized but faster queries.

### API Architecture

- We chose flask over direct DB access for security/scalability.

### Visualization

- We used chart.js (CDN) for its balance of simplicity and interactivity.

## 5. Future Enhancements

1. User Authentication
   - Secure access to transaction data.
2. More Advanced Analytics
   - Predictive spending trends, etc
3. Deployment
   - Cloud hosting (e.g., AWS/Azure).

## 6. Conclusion

This project demonstrates our knowledge of end-to-end data processing and visualization skills. The dashboard is production-ready and extensible for real-world financial analysis.

### Deliverables Submitted

- Python scripts (`process_sms.py`, `app.py`).
- SQLite database + schema.
- Interactive frontend (`index.html`).

### Link to github

https://github.com/d-k0de/MoMo_Data_Analysis