# Starlogic-CDR-Analytics

A Telecom Analytics Solution developed to identify, track, and measure the use of a callback service for prepaid users with critically low airtime

## 📡 Starlogic - Missed Call Trigger Analytics for Telcos

\*\*Starlogic\*\* is a telecom intelligence system that analyzes missed call triggers used by low-balance customers to prompt call-backs—transforming micro-interactions into measurable revenue streams. This project extracts and validates missed call events from Call Data Records (CDRs), classifies them (ON\_NET / OFF\_NET), and attributes revenue accordingly for vendor settlements.

---

## 🚀 Project Objective

To identify and analyze \*intentional missed call triggers\* used by subscribers with ≤ ₦5 airtime, by:

- Detecting flash calls (via `8850`-prefixed numbers)

- Validating response calls made within 1 hour

- Tagging ON\_NET vs OFF\_NET responses

- Calculating Minutes of Use (MoU) & Revenue

- Powering monthly revenue sharing at an 80-20 split with vendors

---

## 🧠 Business Logic

- \*\*Eligibility:\*\* Customers with ≤ ₦5 airtime can use the service

- \*\*Trigger:\*\* Customer initiates call to a number prefixed with `8850`

- \*\*Validation:\*\* Receiver must return the call within \*\*1 hour\*\*

- \*\*Pairing Logic:\*\* When multiple flash calls exist, the response is matched based on chronological proximity

- \*\*Service Classes:\*\* Primarily available to \*prepaid service classes\* only

---

## 🛠️ Tech Stack

- \*\*SQL (DB2 dialect)\*\* – For querying CDRs and modeling pairing logic

- \*\*Python (optional)\*\* – For future ETL automation or visualization

- \*\*Excel Output (.xlsx)\*\* – For final reporting and vendor payout records

- \*\*GitHub Actions (optional)\*\* – For scheduling daily/weekly spools

---

## 🧾 SQL Modules Breakdown

| Module | Description |

|---------------------------|-------------|

| `starlogic\_onnet.sql` | Identifies ON\_NET return calls within 1 hour |

| `starlogic\_offnet.sql` | Tracks OFF\_NET response calls and durations |

| `pairing\_logic.sql` | Performs 1-to-1 and 1-to-all call pairings |

| `revenue\_calculation.sql` | Computes MoU and splits revenue by account |

---

## 📈 Output Metrics

- `C1`: Total Triggered Calls

- `C2`: Unique Triggering Subscribers

- `C3`: Unique Receivers (called numbers)

- `C4 & C5`: 1:1 ON\_NET Calls + Unique Count

- `C6 & C7`: 1:1 OFF\_NET Calls + Unique Count

- MoU Distribution across DA accounts

---

## 📊 Daily and Monthly Reporting

- Data is spooled daily based on call logs

- MoU and user counts are aggregated by date

- Final Excel file is generated monthly for:

- Vendor payout processing

- Service usage analysis

- Compliance and QA

---

## 🧠 Future Enhancements

- 📦 Package as a Python ETL pipeline

- 📊 Integrate with Tableau/Power BI for dashboards

- ☁️ Migrate to AWS Athena or BigQuery for scalability

- 🔐 Add anomaly detection for fraudulent pairing patterns

---

## 🤝 Contribution

Pull requests welcome! For major changes, please open an issue first to discuss what you'd like to change.

---

## 📜 License

[MIT](./LICENSE)

---

## 🙌 Acknowledgements

Built by \*\*Olawale Falodun\*\*, a Data Engineer & Business Intelligence Analyst, with real-world understanding of telco customer behavior and micro-revenue engines.

[LinkedIn](https://www.linkedin.com/in/olawale-falodun-b26b61ab/)

> "You'd be amazed how much ₦5 can trigger, if the call back is strategic."

---

## 🚀 How to Use

1. Clone the repository

```bash

git clone https://github.com/yourusername/starlogic-cdr-analytics.git

2. Navigate to the SQL directory and run queries:

- Start with create\_tables.sql

- Proceed with starlogic\_onnet.sql and starlogic\_offnet.sql

- Finish with da\_usage\_summary.sql

3. Update date ranges inside queries for the desired period.

4. Export results to your BI tool or Excel for reporting.

---